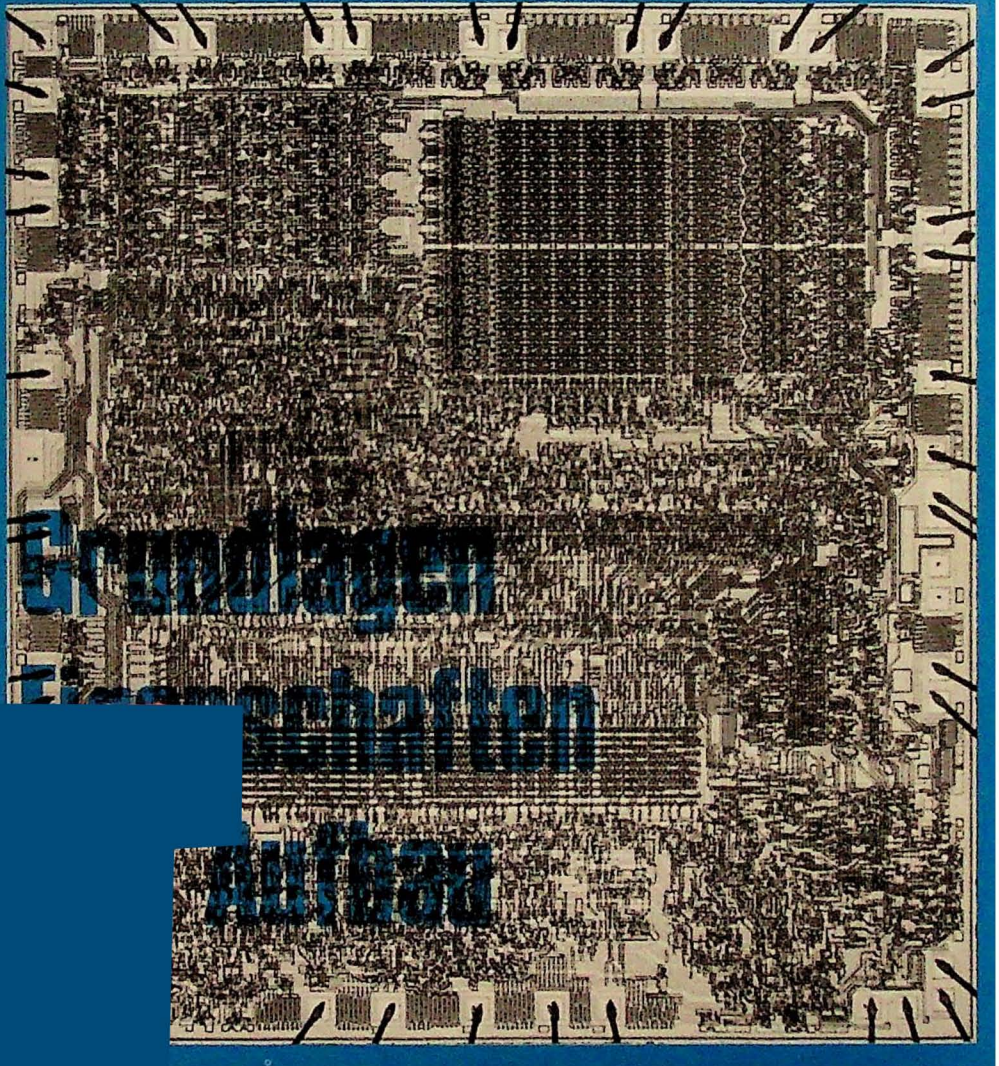
H. Bernstein

MIKROPROZESSOR



ISBN 3-921682 - 22 - 3

2. völlig neu Überarbeitete Auflage. 1978

Es kann keine Gewähr übernommen werden, daß die in diesem Buche verwendeten Angaben Schaltungen, Warenzeichen, Warenbezeichnungen und Programme frei von Schutzrechten Dritter sind. Alle Angaben werden nur für Amateurzwecke mitgeteilt. Alle Daten und Vergleichsangaben sind als unverbindliche Hinweise zu verstehen. Sie geben in keinem Falle Aufschluß über Verfügbarkeit und Liefermöglichkeit. Auf jeden Fall sind die Datenblätter der Hersteller heranzuziehen. Nachdruck und öffentliche Wiedergabe verboten. Irrtum, sowie alle Rechte vorbehalten.

Copyright by Ing.W.Hofacker GMBH , Postfach 75437 , D-8000 München 75

Gedruckt in der Bundesrepublik Deutschland - Printed in Western Germany- Imprimé en RFA

ING. W. HOFACKER

GMBH

FACHBUCHVERLAG · ELEKTRONIK
8 MÜNCHEN 75
POSTFACH 437
Tel. 08024/7331

Vorwort

Seit geraumer Zeit werden von verschiedenen Halbleiter-Herstellern Mikroprozessor-Systeme angeboten. Durch die Einführung von Mikroprozessoren kam eine neue Technologie und Terminologie nicht nur für den Hersteller, sondern auch für den Anwender. Heute finden wir bereits auf dem Halbleitermarkt komplette Systeme auf einem Halbleiter-Chip.

Was verbirgt sich hinter dem Wort „Mikroprozessor"? Nach der Entwicklung des Transistors und der Planartechnik kam die Integration von kompletten Schalteinheiten. Der Entwickler kann an dieser integrierten Schalteinheit nichts verändern, da das System von den Halbleiter-Herstellern vorgegeben wird. Er hat aus dieser System-Vorgabe sein spezielles Programm heraus zu entwickeln. Für die Erstellung des Programmes, der Software, liegen genaue Instruktionen vor:
a) Genaue Bestimmung der Problemdefinition
b) Bestimmung des Lösungsweges
c) Entwicklung des Flußdiagrammes
d) Die Übersetzung in die Programmiersprache
e) Die Richtigkeit mit Hilfe des Programmtestes zu untersuchen
Durch diese Instruktionen ist der Entwickler gezwungen, sein Problem nicht „Hardware" mäßig mit dem Lötkolben zu lösen, sondern durch die Entwicklung eines Programmes, der „Software".

Dieses Buch beschäftigt sich mit dem Aufbau von Mikrocomputern und Mikroprozessoren von der Hard- und Software aus.

Der Autor möchte sich bei folgenden Firmen für die großzügige Unterstützung bedanken:
Texas Instruments, München-Freising
Siemens, München
Intel, München

Der Autor, Herbert Bernstein, Jahrgang 1946, arbeitete nach dem Studium der Elektronik (mit Datenverarbeitung) in einem Entwicklungslabor für elektronische Geräte. 1971 erschien bei der Firma PEK (Dr. Paul E. Klein) sein erstes Handbuch über die Kernspeicher-Technik und digitale und analoge Elektronik. Für die Fernschule in Bremen ist er jetzt als Fachbereichsleiter für die Analog- und Digitaltechnik und als Lehrgangsleiter für das Elektronik-Labor zuständig. Heute arbeitet er als freier Fachschriftsteller und als Dozent am Münchner Elektronik Zentrum.

H. Bernstein

Inhaltsverzeichnis

		Seite
1.	Einführung	
1.1	Einleitung	1
1.2	Mensch und Computer	1
1.3	Aufbau einer Rechenanlage	3
2.	Grundlagen von EDV-Anlagen	
2.1	Code-Maschinensprache von Rechenanlagen	5
2.2	Codieren von Zahlen	6
2.3	Flußdiagramme	11
2.4	Einadreßmaschine	13
2.5	Addition	15
2.6	Subtraktion	18
2.7	Multiplikation	20
2.8	Division	21
2.9	Quadrieren	24
3.	Mikrocomputer	
3.1	Ein-Chip-Rechner	26
3.2	Innenschaltung eines Ein-Chip-Rechners	29
4.	Mikroprozessoren Typenzusammenstellung	
4.1	SBA Mikroprozessor	42
4.2	2 BIT-STTL Mikroprozessor	44
4.3	4 BIT Mikroprozessor - Slice	46
4.4	4 BIT ECL Mikroprozessor	48
4.5	4 BIT STTL Mikroprozessor Slice	50
4.6	4 BIT IIL Mikroprozessor	52
4.7	Bipolares 4 BIT Mikroprozessor Slice	54
4.8	Schottky - TTL 4 BIT Mikroprozessor Slice	56
4.9	Macrologic 4 BIT Mikroprozessor	58
4.10	4 BIT Mikroprozessor in CMOS, NMOS u. PMOS	60
4.11	4 BIT PMOS Mikroprozessor	62
4.12	4 BIT Mikroprozessor in PMOS	64
4.13	8 BIT Mikroprozessor in NMOS 8080	66
4.14	8 BIT Mikroprozessor in NMOS 650X	68
4.15	8 BIT Mikroprozessor in NMOS 6800	70
4.16	8 BIT Mikroprozessor in NMOS Z 80	72
4.17	8 BIT Mikroprozessor in PMOS 8008	74
4.18	8 BIT Mikroprozessor in NMOS 8085	76
4.19	8 BIT Mikroprozessor in NMOS SC/MP	78
4.20	8 BIT Mikroprozessor in NMOS 3850	80
4.21	8 BIT Mikroprozessor in NMOS 6802	82
4.22	12 BIT Mikroprozessor in CMOS IM 6100	84
4.23	16 BIT Mikroprozessor in NMOS mN 601	86
4.24	16 BIT Mikroprozessor in NMOS CP 1600	88
4.25	16 BIT Mikroprozessor in NMOS oder PMOS	90
4.26	16 BIT Mikroprozessor in NMOS und IIL TMS 9900	92
5.	Aufbau eines Mikroprozessor-Systems	94

6. Mikroprozessor-System 6800
6.1 MPU 6800 96
6.2 Taktgenerator für den Mikroprozessor 6800 99
6.3 Der Schreib-Lese-Speicher (RAM) 99
6.4 Der Programm-Speicher (ROM) 99
6.5 Kurzbeschreibung eines peripheren Interface-Adapters (PIA) . . . . . ,100
6.6 Kurzbeschreibung eines asynchronen Interface-Adapters (ACIA) 104
6.7 Kurzbeschreibung eines synchronen seriellen Daten-Adapters (SSDA) 104
6.8 Kurzbeschreibung eines Modulators und Demodulators (MODEM) 107
6.9 Kurzbeschreibung eines universellen synchronen/asynchronen Empfänger/Sender-Bausteines (USART) 107

7. Der Mikroprozessor 8080
7.1 Einführung 111
7.2 Aufbau des 8080 111
7.2.1 Register-Matrix und Adressen-Logik 111
7.2.2 Arithmetik-Logik-Einheit (ALU) 114
7.2.3 Befehls-Register und Steuer-Logik 115
7.2.4 Verbindungen im Mikroprozessor 116
7.3 Arbeitszyklus des 8080 117
7.4 Befehlszyklus der 8080 119
7.5 Reihenfolge der Operationsschritte 123
7.6 Operationsschritte in der 8080 124
7.7 Ablauf bei Programmunterbrechungen 126
7.8 Direkter Speicherzugriff bei der 8080 129
7.9 HALT-Ablauf bei der 8080 130
7.10 Einschalten des 8080 132

1. Einführung

1.1. Einleitung

Die Entwicklung der elektronischen Datenverarbeitungsanlagen begann mit der Fertigstellung des ersten mechanischen Rechenwerkes. Dem Konstrukteur WILHELM SCHICKARD gelang im Jahre 1623 die erste mechanische Addition und Subtraktion von Zahlen. Mit diesem mechanischen Computer wurden sechsstellige Zahlen addiert und subtrahiert.

1671 wurde von dem Mathematiker G.W. LEIBNITZ das duale Zahlensystem entwickelt. Kurze Zeit später war die theoretische Addition, Subtraktion, Multiplikation und Division mit mechanischen Rechenanlagen gelöst. Erst hundert Jahre später konnten die theoretischen Rechenbeispiele durchgeführt werden.

Die ersten Rechenanlagen für die Programmsteuerung für Webanlagen erfand JAQUARD im Jahre 1804. Sein Landsmann BABBAGE nahm die Idee auf und entwickelte die Lochkarte für die Programmsteuerung. Mit Hilfe von Lochkarten konnte nun eine Programmsteuerung für Webstühle mechanisch bestimmt und geändert werden. 1821 baute BABBAGE den ersten mechanischen Prozeßrechner, den er 1848 wesentlich verbesserte. Diese Anlage war in der Lage, mathematische Probleme mechanisch zu lösen. Die Grundidee, Rechenoperationen, Datenspeicherung und Programmsteuerung nur von einem Gerät durchführen zu lassen, scheiterten an den mechanischen Hilfsmitteln.

Ab 1850 entstanden die einfachen Lochkartenmaschinen auf der Basis mechanischer Grundlagen. Durch die Verbesserung der mechanischen Hilfsmittel wurde die Bedienung vereinfacht und die Rechengeschwindigkeit erhöht. Wesentlich zur Erhöhung der Rechengeschwindigkeit trug die Erfindung des Elektromotors bei. Elektromotoren betrieben nun die mechanischen Rechenanlagen.

H. HOLLERITH entwickelte Ende des 19. Jahrhunderts die erste praktisch verwendbare Lochkartenmaschine. 1890 wurde diese Maschine bei der Volkszählung in den USA eingesetzt.

Mit der Einführung der Elektronen-Röhre begann der Bau der ersten elektrischen Rechenanlagen. In den dreißiger Jahren wurden die Ideen von BABBAGE und CAUFFIGNAL aufgefaßt und es entstand die erste Konzeption für die heutigen Rechenanlagen. 1935 wurde von ZUSE der erste Relais rechner vorgestellt. Der aus lauter Relais bestehende Rechner arbeitete im Dualsystem.

Unter der Leitung des Mathematikers AIKEN entsteht 1944 ein programmgesteuerter Relaisrechner. 1946 arbeitet der erste vollelektronische Rechenautomat unter der Leitung von GOLDSTINE, ECKERT und MAUCHLY.

Mit der Erfindung des Transistors und der integrierten Schaltung wurde das gesamte Rechnersystem vereinfacht und rationalisiert. Heute wird der Computer von der Hausfrau ebenso verwendet, wie vom Universitätsprofessor.

1.2. Mensch und Computer

Im Bild 1.2.1 und Bild 1.2.2 ist die Beziehung zwischen Mensch und Maschine gezeigt. Auf den Menschen wirkt die Problemstellung ein. Was ist ein Problem? Sämtliche Umwelteinflüsse stellen für den Menschen Probleme dar. Er unterscheidet nur zwischen dem großen und kleinen Problem.
Das gestellte Problem versucht der Mensch zu verarbeiten und erzeugt daraus seine Handlung und seine Handlungsweise. Um eine eigene Handlungsweise zu entwickeln benötigt er sein Gehirn, also ein Gedächtnis. Reicht sein Gedächtnis, sein Erlerntes nicht mehr aus, greift er zur Fachliteratur und zu den Tabellen. Er informiert sich. Im Gedächtnis und im Gehirn fällt die Entscheidung. Notfalls verwendet er Hilfsmittel, z.B. den Rechenschieber.
Nach einer Berechnung gibt er die Handlung frei. Arbeitet der Mensch richtig, ist sein Gedanke, seine Handlung richtig.

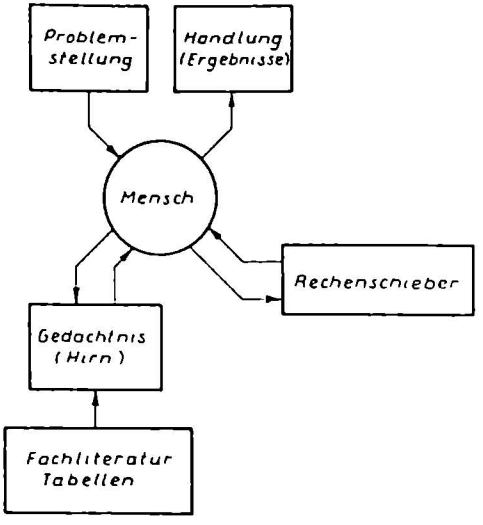


Bild 1.2.1 Handlungsaufbau beim Menschen

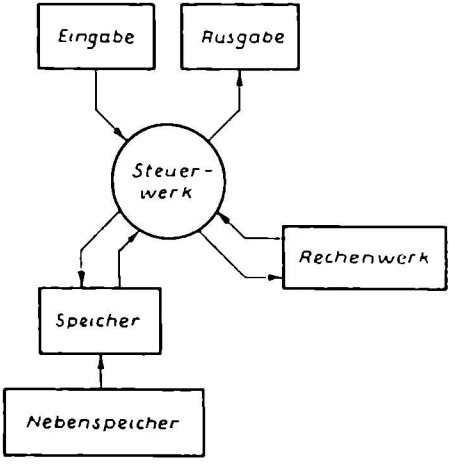


Bild 1.2.2 Handlungsaufbau bei Rechenmaschinen

Vergleicht man den Aufbau einer Datenverarbeitungsanlage mit dem Menschen, so ist im wesentlichen der Aufbau identisch. Das Steuerwerk erhält über die Eingabe die zu verarbeitenden Daten oder Informationen. Das Steuerwerk kann den Speicher abfragen, ob eine ähnliche Information vorliegt oder es kann aus dem Speicher einen Befehl erhalten. Reichen die gespeicherten Informationen im Speicherwerk nicht aus, so können durch einen Nebenspeicher weitere Informationen auf das Steuerwerk gegeben werden. Die Verarbeitung von Informationen geschieht im Rechenwerk. Die Ausgabe von Daten und Informationen erfolgt über die Ausgabe.
Das Steuerwerk kann nur dann richtig arbeiten, wenn ein richtiges Programm vorliegt. Der Programmierer hat die Aufgabe, das Steuerwerk über den Speicher richtig zu programmieren, da
a) ein Steuerwerk keine Probleme lösen kann, die nicht programmiert sind,

b) ein Steuerwerk keine unvorhergesehenen Entscheidungen treffen kann, die nicht programmiert sind,
c) ein Steuerwerk noch keine eigenen Programme schreiben oder entwickeln kann und
d) ein Steuerwerk nur mit Programmbegriffen und seiner eigenen Symbolsprache umgehen kann.

1.3. Aufbau einer Rechenanlage

Im Bild 1.3.1 ist der Aufbau einer Rechenanlage gezeigt. Eine Rechenanlage besteht im wesentlichen aus fünf Teilen.

Eingabe: Über die Eingabe erfolgt das Einschreiben der Information. Die Eingabe der Informationen kann über eine Tastatur, einen Lochstreifenleser, einen Fernschreiber, einen elektronischen Empfänger oder über andere Einrichtungen erfolgen.

Speicher: Die Daten und Informationen der Eingabe werden hier gespeichert. Die Speicherung ist eine der wichtigsten Kriterien in den Rechenanlagen. Vom Speicher aus werden die Daten an das Rechenwerk weitergegeben. Das Rechenwerk gibt seine Ausgangsdaten an den Speicher zurück. Die verarbeiteten Informationen können durch den Speicher zwischengespeichert um danach auf die Ausgabe gegeben zu werden.

Rechenwerk: Das Rechenwerk ist der Mittelpunkt in einer Rechenanlage. Hier werden die Eingangsinformationen des Speichers verarbeitet. Die Ausgangsinformationen werden auf den Speicher zurückgegeben. Rechenwerke mit arithmetischen und logischen Aufgaben haben die Kurzbezeichnung „ALU" (Arithmetik-Logik-Einheit).

Ausgabe: Über die Ausgabe erfolgt die Daten- und Informationsausgabe aus einer Rechenanlage. Als Ausgabe kann ein Fernschreiber, ein Lochstanzer, ein Datensichtgerät, ein Blattschreiber, Drucker, ein elektronischer Sender oder eine andere Einrichtung sein.

Steuerwerk: Mit dem Steuerwerk kann der gesamte Ablauf in der Anlage gesteuert werden. Über das Steuerwerk erhält die Eingabe den Befehl zur Datenaufnahme. Danach gibt das Steuerwerk den Befehl „Speicher übernimm die Eingangsdaten und speichere sie". Durch das Steuerwerk bekommt der Speicher und das Rechenwerk den Befehl „Speicher gibt Daten an Rechenwerk und Rechenwerk addiere". Der Speicher gibt seine Daten an das Rechenwerk, wo diese dann addiert werden. Durch einen Befehl an den Speicher „Speicher nimm das Ergebnis des Rechenwerks auf und speichere es", werden die Ausgangsdaten des Rechenwerkes zwischengespeichert. Durch den Befehl „Speicher lies aus" erhält die Ausgabe seine zwischengespeicherten Informationen auszulesen.

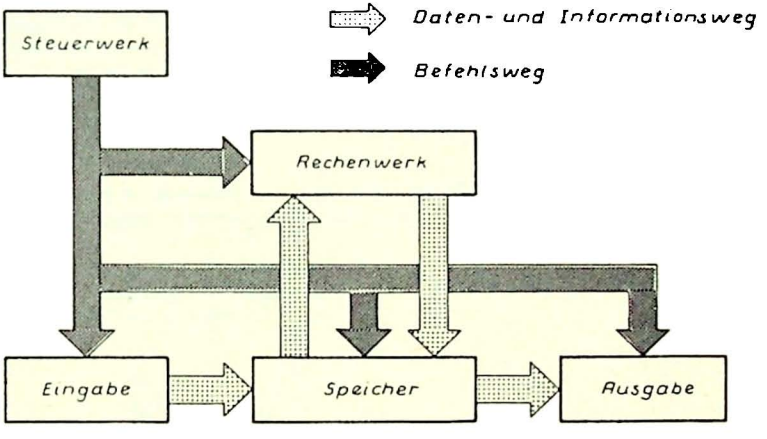


Bild 1.3.1 Aufbau einer Rechenanlage

In einem Rechenwerk unterscheidet man zwischen dem Daten- und Informationsweg und dem Befehlsweg. In dem Datenweg werden die Informationen zu den einzelnen Stationen in der Rechenanlage transportiert. Der Befehlsweg liegt praktisch an allen Funktionen der Rechenanlage an. Über den Be-

fehlsweg erhält jede Einheit im Rechenwerk seinen entsprechenden Befehl. Typische Befehle sind:

„Speicher laden mit Inhalt der Eingabe"
„Speicherinhalt an Ausgabe geben"
„Speicherinhalt an Rechenwerk"
„Ergebnis vom Rechenwerk in Speicher laden".

2. Grundlagen von EDV Anlagen

2.1. Code-Maschinensprache von Rechenanlagen

Der Mathematiker LEIBNITZ entwickelte im 17. Jahrhundert das duale Zahlensystem, das Rechnen mit 0 und 1. Jede Zahl kann durch eine 0 und eine 1 dargestellt werden. Es gibt sich daraus folgende Tabelle:

Dezimale Darstellung	Duale Darstellung (alte)	(neue)	(neueste)
0	0	0	L
1	1	L	H
2	10	L0	HL
3	11	LL	HH
4	100	L00	HLL
5	101	L0L	HLH
6	110	LL0	HHL
7	111	LLL	HHH
8	1000	L000	HLLL
9	1001	L00L	HLLH
10	1010	L0L0	HLHL
11	1011	L0LL	HLHH
12	1100	LL00	HHLL
13	1101	LL0L	HHLH
14	1110	LLL0	HHHL
15	1111	LLLL	HHHH
16	10000	L0000	HLLLL

Das duale Zahlensystem wird in drei Formen dargestellt:

0,1: Die alte Darstellungsform führte zu leichten Verwechslungen mit dem Dezimalsystem.

0,L: Diese Form der Darstellung wird in den meisten Lehrbüchern gebracht.

L,H: In der modernen Elektronik findet man nur noch diese Darstellungsweise. Mit L bezeichnet man keine Spannung, keinen Strom, also kein Signal. Mit L für low bezeichnet man nur Signalpegel. Unter der Spannung von 2V hat man z.B. ein L-Signal in der TTL-Technik. Bei einem Signalpegel von über 2V hat man ein H-Signal. Mit H für high bezeichnet man hohe Signalpegel.

Bei den Mikroprozessoren verwendet man noch eine Darstellungsart von Zahlen, die sedezimale Darstellung. Die nachfolgende Tabelle zeigt alle drei Arten:

Dezimale Darstellung	Duale Darstellung	Sedezimale Darstellung
0	L	0
1	H	1
2	HL	2
3	HH	3
4	HLL	4
5	HLH	5
6	HHL	6
7	HHH	7
8	HLLL	8
9	HLLH	9
10	HLHL	A

11	HLHH	B
12	HHLL	C
13	HHLH	D
14	HHHL	E
15	HHHH	F
16	HLLLL	10
17	HLLLH	11

Mit dem sedezimalen Zahlensystem ist man in der Lage, den gesamten Darstellungsbereich der dualen Zahlen zu erfassen. Mit dem dezimalen Zahlensystem können wir nur die Zahlen von 0 bis 9 des dualen Zahlensystems erfassen. Die dualen Zahlen zwischen 10 und 15 bezeichnet man als Pseudotetraden. Diese Pseudotetraden entfallen in der sedezimalen Darstellung. Die Zahlen zwischen 10 und 15 sind mit Buchstaben gekennzeichnet.

2.2. Codieren von Zahlen

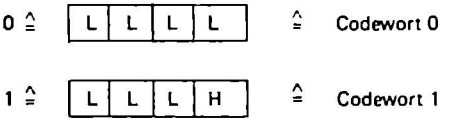
Durch das Codieren von Zahlen werden dezimale Zahlen in die Maschinenzahlen umgewandelt.

0	0	LLLL
1	1	LLLH
2	2	LLHL
3	3	LLHH
4	4	LHLL
5	5	LHLH
6	6	LHHL
7	7	LHHH
8	8	HLLL
9	9	HLLH
10	A	HLHL
11	B	HLHH
12	C	HHLL
13	D	HHLH
14	E	HHHL
15	F	HHHH

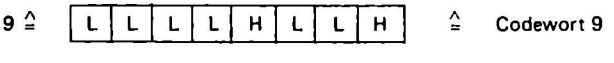
Beispiele:

Die Dezimalzahl 1 besteht aus einer Binärstelle, 1 = H.
Die Dezimalzahl 3 besteht aus zwei Binärstellen, 3 = HH.
Die Dezimalzahl 5 besteht aus drei Binärstellen, 5 = HLH.
Die Dezimalzahl 9 besteht aus vier Binärstellen, 9 = HLLH.

Jede Binärstelle bedeutet ein Bit, d.h. wenn wir die Dezimalzahl 9 darstellen wollen, benötigen wir vier Bits. Die Aneinandersetzung von mehreren Bits bezeichnet man als Codewort.



Vierstellige Codewörter finden wir in der Elektronik nur sehr selten. Besonders häufig sind dagegen die achtstelligen Codewörter, die sogenannten „Bytes".



Mit einem Byte können 256 Informationen dargestellt werden.

Auflösung eines Bytes in Potenzschreibweise:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Mit einem Byte können 2^8 Informationen dargestellt werden. Für die Datenverarbeitung bedeutet das, in einem Byte können neben Zahlen und Buchstaben auch Sonderzeichen dargestellt werden.
Ein typischer Byte-Code ist der American Standard Code for Information Interchange (Kurzbezeichnung ASCII-Code).

	ASCII								
	P	b_7	b_6	b_5	b_4	b_3	b_2	b_1	ASCII SYMBOL
0	H	L	H	L	L	L	L	L	Space
1	H	L	H	H	L	L	L	H	1
2	H	L	H	H	L	L	H	L	2
3	L	L	H	H	L	L	H	H	3
4	H	L	H	H	L	H	L	L	4
5	L	L	H	H	L	H	L	H	5
6	L	L	H	H	L	H	H	L	6
7	H	L	H	H	L	H	H	H	7
8	H	L	H	H	H	L	L	L	8
9	L	L	H	H	H	L	L	H	9
10	L	L	H	H	L	L	L	L	0
11	H	L	H	L	L	L	H	H	=
12	H	H	L	L	L	L	L	L	@
13	L	L	H	H	H	L	H	L	:
14	H	L	H	H	H	H	H	L	>
15	L	L	H	H	H	H	H	H	?
16	H	H	L	H	H	L	H	H	[
17	H	L	H	L	H	H	H	H	/
18	L	H	L	H	L	L	H	H	S
19	H	H	L	H	L	H	L	L	T
20	L	H	L	H	L	H	L	H	U
21	L	H	L	H	L	H	H	L	V
22	H	H	L	H	L	H	H	H	W
23	H	H	L	H	H	L	L	L	X
24	L	H	L	H	H	L	L	H	Y
25	L	H	L	H	H	L	H	L	Z
26	L	L	L	L	H	L	H	L	LF
27	H	L	H	L	H	H	L	L	,
28	H	L	H	L	L	H	L	H	%
29	L	L	L	L	H	L	L	H	HT
30	L	L	H	L	L	H	H	H	'
31	L	L	H	L	L	L	H	L	"
32	L	L	H	L	H	H	L	H	–
33	H	H	L	L	H	L	H	L	J
34	L	H	L	L	H	L	H	H	K
35	H	H	L	L	H	H	L	L	L
36	L	H	L	L	H	H	L	H	M
37	L	H	L	L	H	H	H	L	N
38	H	H	L	L	H	H	H	H	O

39	L	H	L	H	L	L	L	L	P
40	H	H	L	H	L	L	L	H	Q
41	H	H	L	H	L	L	H	L	R
42	L	L	H	L	L	L	L	H	!
43	L	L	H	L	L	H	L	L	$
44	H	L	H	L	H	L	H	L	*
45	H	L	H	L	H	L	L	H	)
46	H	L	H	H	H	L	H	H	;
47	H	H	L	H	H	H	L	H	]
48	H	L	H	L	L	H	H	L	&
49	L	H	L	L	L	L	L	H	A
50	L	H	L	L	L	L	H	L	B
51	H	H	L	L	L	L	H	H	C
52	L	H	L	L	L	H	L	L	D
53	H	H	L	L	L	H	L	H	E
54	H	H	L	L	L	H	H	L	F
55	L	H	L	L	L	H	H	H	G
56	L	H	L	L	H	L	L	L	H
57	H	H	L	L	H	L	L	H	I
58	L	L	H	L	H	L	H	H	+
59	L	L	H	L	H	H	H	L	.
60	H	H	L	H	H	H	H	L	¬
61	L	L	H	L	H	L	L	L	(
62	L	L	H	H	H	H	L	L	<
63	H	L	L	L	H	H	L	H	CR

Den ASCII-Code können wir nach mehreren Gesichtspunkten überprüfen:

a) Die ranghöchste Spur ist die Prüfbit-Spur P oder b_8. Diese Spur kann von einer Angabe-Einheit erzeugt oder elektronisch durch eine Komparatorschaltung hergestellt oder direkt durch einen Codierer erzeugt werden.
   Für den ASCII-Code gilt:
   P = H, wenn b_1 bis b_7 ungeradzahlige Bitwerte aufweisen, 1, 3, 5, 7
   P = L, wenn b_1 bis b_7 geradzahlige Bitwerte aufweisen, 0, 2, 4, 6
   Diese Spur dient nur für Prüfzwecke bei der Übermittlung.

b) Die fünf Spuren b_1 bis b_5 sind die Zeichenspuren. Durch die Veränderung der fünf Spuren können die Symbole übertragen werden.

c) Die zwei Spuren b_6 und b_7 dienen zur Umschaltung zwischen den Zahlen, Buchstaben und Sonderzeichen.
   b_6 = b_7 = L: Diese Kombination tritt nur an der Stelle 63 auf und bedeutet CR = carriage return = Wagenrücklauf. Normalerweise können bei dieser Bitanordnung spezielle Sonderfunktionen übertragen werden, die aber in der Codetabelle nicht aufgeführt sind. Wir können drei Befehlsarten übermitteln:
   CC-Befehl = communication controll = Übertragungs-Steuer-Befehl
   FE-Befehl = format effector = Format-Steuer-Befehl
   IS-Befehl = information separator = Trennbefehl
   Durch diese Erweiterung gibt es noch 32 Zusatzbefehle:

b_5	b_4	b_3	b_2	b_1				
L	L	L	L	L	NL		Null	null (zero)
L	L	L	L	H	SOM	CC-Befehl	Start der Nachricht	start of message
L	L	L	H	L	STX	CC-Befehl	Start des Textes	start of text
L	L	L	H	H	ETX	CC-Befehl	Ende des Textes	end of text
L	L	H	L	L	EOT	CC-Befehl	Ende der Übertragung	end of transmission
L	L	H	L	H	ENQ	CC-Befehl	Wer da? (Anruf)	enquiry
L	L	H	H	L	ACK	CC-Befehl	Bestätigung	acknowledge
L	L	H	H	H	BEL	CC-Befehl	Klingel	bell

L	H	L	L	L	BS	FE-Befehl	Rückschritt	back space
L	H	L	L	H	HT	FE-Befehl	horizontale Tabellierung	horizontal tabulation
L	H	L	H	L	LF	FE-Befehl	Zeilenvorschub	line feed
L	H	L	H	H	VT	FE-Befehl	vertikale Tabellierung	vertical tabulation
L	H	H	L	L	FF	FE-Befehl	Formularvorschub	form feed
L	H	H	L	H	CR	FE-Befehl	Wagenrücklauf	carriage return
L	H	H	H	L	SO	FE-Befehl	Wagen ausrücken	shift out
L	H	H	H	H	SI	FE-Befehl	Wagen einrücken	shift in
H	L	L	L	L	DLE	CC-Befehl	Unvollständige Daten	data link escape
H	L	L	L	H	DC1	CC-Befehl	Steuersignal 1	device control 1
H	L	L	H	L	DC2	CC-Befehl	Steuersignal 2	device control 2
H	L	L	H	H	DC3	CC-Befehl	Steuersignal 3	device control 3
H	L	H	L	L	DC4	CC-Befehl	Steuersignal 4	device control 4
H	L	H	L	H	NAK	CC-Befehl	Falschmeldung	negative acknowledge
H	L	H	H	L	SYN	CC-Befehl	Synchroner Betrieb	synchronous idle
H	L	H	H	H	ETB	CC-Befehl	Ende des Übertragungs-blockes	end of transmission block
H	H	L	L	L	CAN	CC-Befehl	Widerruf	cancel
H	H	L	L	H	EM	CC-Befehl	Ende des Mediums	end of medium
H	H	L	H	L	SS	CC-Befehl	Start einer Spezialfolge	start of special sequence
H	H	L	H	H	ESC	CC-Befehl	entkommen	escape
H	H	H	L	L	FS	IS-Befehl	Trennzeichen für Abschnitt	file separator
H	H	H	L	H	GS	IS-Befehl	Trennzeichen für Gruppe	group separator
H	H	H	H	L	RS	IS-Befehl	Trennzeichen für Auf-zeichnung	record separator
H	H	H	H	H	US	IS-Befehl	Trennzeichen für Einheit	unit separator

b_7 = L und b_6 = H: Bei dieser Kombination werden die Sonderzeichen und Zahlen geschrieben, z.B. !, ¢, ? usw. Hier nimmt nur der Befehl „SP" für „space"=Zwischenraum eine Sonderstellung ein. Nach Eingeben des SP-Befehls wird zwischen zwei Abschnitten ein Platz freigelassen.

b_7 = H und b_6 = L: Bei dieser Kombination werden die großen Buchstaben geschrieben. Ebenso befinden sich sechs Sonderzeichen auf der Kombination. Diese Zeichen werden aber nur sehr selten benötigt.

b_7 = H und b_6 = H: Bei dieser Kombination werden die kleinen Buchstaben geschrieben. Ebenso befinden sich fünf Sonderzeichen und ein besonderer Befehl auf dieser Kombination. Der besondere Befehl ist der DEL-Befehl. Vertippen wir uns beim Programmieren und stellen den Fehler erst nach der Eingabe von weiteren Befehlen fest, so kann mit dem DEL-Befehl der Fehler gelöscht oder ausgestrichen werden. Mit dem BS-Befehl müssen wir die Fehlerstelle erreichen, d.h. liegt der Fehler z.B. 10 Schritte weiter zurück, so muß der BS-Befehl 10mal kommen. Nach der Fehlerkorrektur muß der DEL-Befehl 10mal kommen.
Bei einer zurückliegenden Fehlerstelle muß zuerst DEL-Befehl gegeben werden, d.h. DEL $\hat{=}$ streichen $\hat{=}$ delete. Danach müssen wir schrittweise auf die Fehlerstelle zurückgehen. Dabei geben wir je Schritt einen BS-Befehl ab. Bei Erreichen der Fehlerstelle wird der Fehler korrigiert, d.h. wir geben die richtige Information ein. Durch den DEL-Befehl können wir uns nach der Korrektur dem geschriebenen Ausgangspunkt schrittweise nähern, d.h. pro Schritt ein DEL-Befehl.

Ein weiterer Byte-Code ist der EBCDIC-Code (Extended Binary Coded Interchange Code):

	EBCDIC								
	0	1	2	3	4	5	6	7	EBCDIC SYMBOL
0	L	H	L	L	L	L	L	L	Space
1	H	H	H	H	L	L	L	H	1
2	H	H	H	H	L	L	H	L	2

3	H	H	H	H	L	L	H	H	3
4	H	H	H	H	L	H	L	L	4
5	H	H	H	H	L	H	L	H	5
6	H	H	H	H	L	H	H	L	6
7	H	H	H	H	L	H	H	H	7
8	H	H	H	H	H	L	L	L	8
9	H	H	H	H	H	L	L	H	9
10	H	H	H	H	L	L	L	L	0
11	L	H	H	H	H	L	H	H	≠
12	L	H	H	H	H	H	L	L	@
13	L	H	H	H	H	L	H	L	:
14	L	H	H	L	H	H	H	L	>
15	L	L	L	H	L	L	H	H	TM
16	L	H	L	L	L	L	L	L	Space
17	L	H	H	L	L	L	L	H	/
18	H	H	H	L	L	L	H	L	S
19	H	H	H	L	L	L	H	H	T
20	H	H	H	L	L	H	L	L	U
21	H	H	H	L	L	H	L	H	V
22	H	H	H	L	L	H	H	L	W
23	H	H	H	L	L	H	H	H	X
24	H	H	H	L	H	L	L	L	Y
25	H	H	H	L	H	L	L	H	Z
26	H	H	H	L	L	L	L	L	RM
27	L	H	H	L	H	L	H	H	,
28	L	H	H	L	H	H	L	L	%
29	L	H	L	L	H	H	H	L	+
30	L	H	L	L	H	L	H	L	¢
31	L	H	H	H	H	H	H	L	=
32	L	H	H	L	L	L	L	L	-
33	H	H	L	H	L	L	L	H	J
34	H	H	L	H	L	L	H	L	K
35	H	H	L	H	L	L	H	H	L
36	H	H	L	H	L	H	L	L	M
37	H	H	L	H	L	H	L	H	N
38	H	H	L	H	L	H	H	L	O
39	H	H	L	H	L	H	H	H	P
40	H	H	L	H	H	L	L	L	Q
41	H	H	L	H	H	L	L	H	R
42	L	H	L	H	H	L	H	L	!
43	L	H	L	H	H	L	H	H	$
44	L	H	L	H	H	H	L	L	*.
45	L	H	L	H	H	H	L	H	)
46	L	H	L	H	H	H	H	L	;
47	L	H	H	H	H	H	H	H	"
48	L	H	L	H	L	L	L	L	&
49	H	H	L	L	L	L	L	H	A
50	H	H	L	L	L	L	H	L	B
51	H	H	L	L	L	L	H	H	C
52	H	H	L	L	L	H	L	L	D
53	H	H	L	L	L	H	L	H	E
54	H	H	L	L	L	H	H	L	F
55	H	H	L	L	L	H	H	H	G
56	H	H	L	L	H	L	L	L	H
57	H	H	L	L	H	L	L	H	I
58	L	H	H	L	H	H	H	H	?
59	L	H	L	L	H	L	H	H	.

60	L	H	H	L	H	L	H	L	¤
61	L	H	L	L	H	H	L	H	(
62	L	H	L	L	H	H	L	L	<
63	L	H	H	H	H	H	L	H	'

2.3. Flußdiagramm

In Bild 2.3.1 sind die Symbole für das Erstellen von Flußdiagrammen gezeigt.

Grenzstelle: Diese Stelle markiert den Beginn, das Ende oder den Zwischenhalt bei einer Rechenoperation.

Ein- oder Ausgabe: Bei der Eingabe geben wir die Befehle in ein Flußdiagramm ein.

Operation: Hier wird der Operationsbefehl eingeschrieben.

Entscheidung oder Verzweigung. Hier kann eine Entscheidung getroffen werden, z.B. a = b, wenn ja oder wenn nein, dann erfolgt jeweils ein bestimmter Befehl.

Übergangsstelle: Kennzeichnung, wenn ein Übergang in ein anderes Programm erfolgt oder wenn die Daten übertragen werden sollen.

Unterprogramm: Hier haben wir die Möglichkeit, das Programm durch ein Unterprogramm, meistens ein Rechenvorgang, zu unterbrechen.

Flußlinie oder Ablaufslinie: Mit den beiden Linien wird der Zusammenhang zwischen den einzelnen Medien hergestellt.

Zusammenführung: Hier erfolgt die Zusammenführung von Fluß- oder Ablaufslinien zu einer gemeinsamen Linie.

Bemerkung: Bei der Bemerkung können wir notwendige Notizen anschreiben. Die Bemerkungen stellen die Legende des Programmes dar.

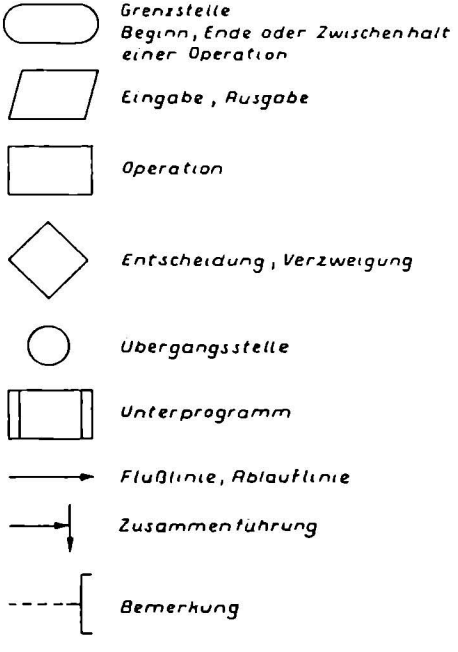


Bild 2.3.1 Symbole für das Erstellen von Flußdiagrammen

In Bild 2.3.2 wird ein Flußdiagramm für den täglichen Morgenablauf gezeigt. Das Programm beginnt. Der Wecker klingelt um 7⁰⁰. Wir sehen aus dem Fenster und treffen eine Entscheidung. Ist das Wetter schön, legen wir uns zurück in das Bett und das Programm ist in diesem Fall beendet. Ist das Wetter aber schlecht, stehen wir auf und knurren. Das Programm gibt uns den Befehl „stellen Kaffeewasser hin". Anschließend lautet der Befehl „geh ins Bad". Da das Kaffeewasser am Herd steht, müssen wir ab und zu horchen, ob das Wasser kocht. Die Entscheidung lautet: „Kocht das Wasser? ". Bei einem

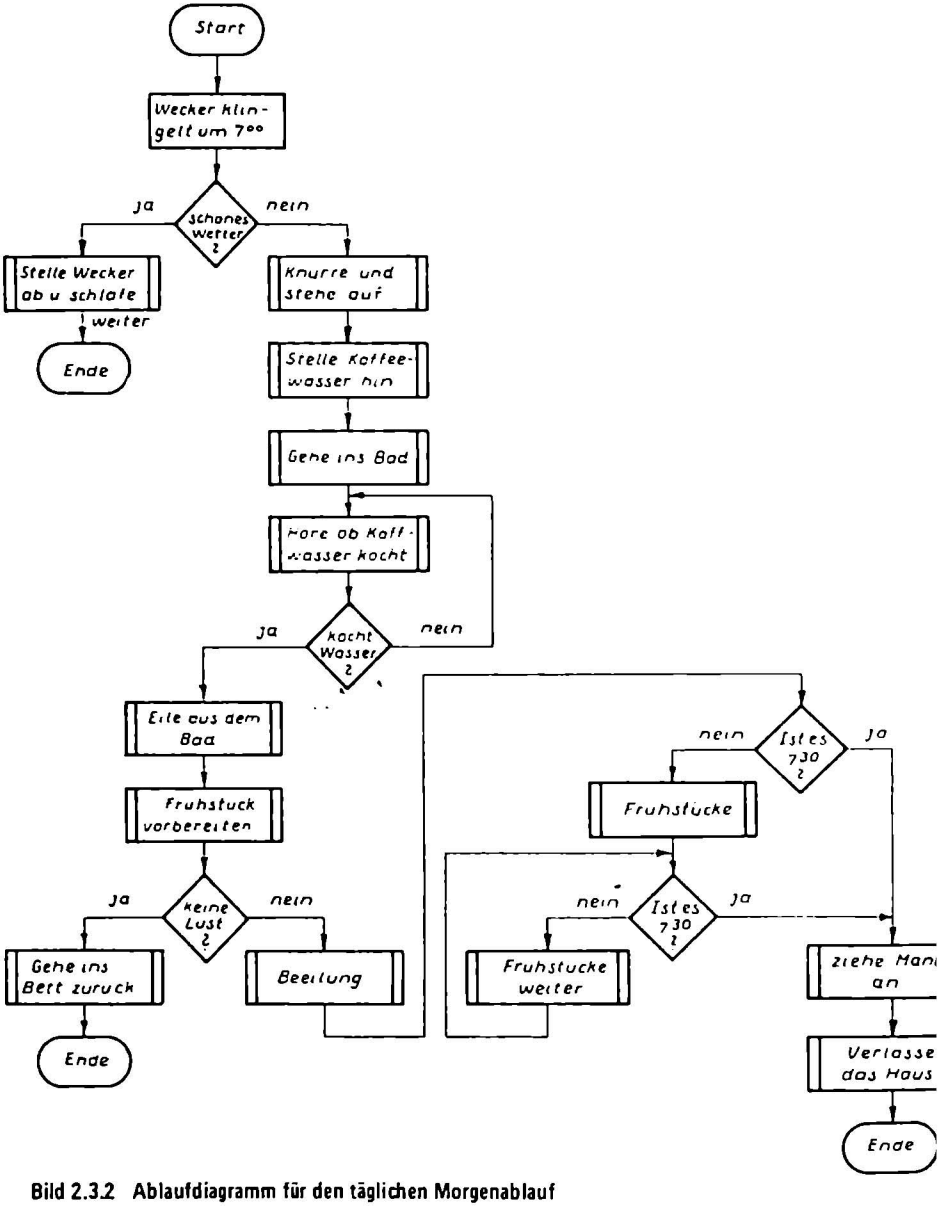


Bild 2.3.2 Ablaufdiagramm für den täglichen Morgenablauf

„nein" befinden wir uns in einem Schleifenprogramm. Wir springen auf den Befehl „höre ob Kaffeewasser kocht" zurück. Diese Schleife durchlaufen wir so oft, bis die Entscheidung auf „ja" lautet. Kocht das Wasser, heißt der nächste Programmschritt „Eile aus dem Bad" und „Frühstück herrichten". Danach erfolgt eine Entscheidung. „Keine Lust? " das Frühstück herzurichten. Bei „ja" gehen wir in unser Bett zurück, bei „nein" „Beeilen" wir uns. Mit der Entscheidung „Ist es 7^{30}? " stellen wir die Zeit fest. Bei „nein", können wir „Frühstücken". Bei „ja" eilen wir in die Diele, „ziehen Mantel an" und „Verlassen das Haus", Programmende.
Ist es noch nicht 7^{30}, können wir „frühstücken". Beim Frühstücken sehen wir auf die Uhr und fragen, „ist es 7^{30}? ". Bei „nein" begeben wir uns auf ein Schleifenprogramm, „frühstücke weiter". Das Schleifenprogramm durchlaufen wir solange, bis wir 7^{30} erreicht haben. Bei „ja" geht der Befehl auf „ziehe Mantel an" und „verlasse das Haus". Das Programm ist beendet.

Flußdiagramme für Rechenanlagen sind wesentlich anders aufgebaut als das Beispiel in Bild 2.3.2. In Bild 2.3.3 ist ein Flußdiagramm für einen Vergleicher gezeigt. Mit dem Start beginnt der Ablauf. Der Rechner erhält den Befehl „Lese Adresse 1 (ADD 1)". Danach den Befehl „Lese Adresse 2 (ADD 2)". In der Entscheidung wird nun der Inhalt der beiden Adressen miteinander verglichen. Ist die ADD 1 größer als die ADD 2, so wird die ADD 1 in den Speicherplatz SP 3 abgespeichert. Ist aber ADD 1 kleiner als die ADD 2, so wird die ADD 1 in den Speicherplatz SP 4 gespeichert.
In dem Flußdiagramm erfolgt die Speicherung der Adresse ADD 1 nach dem Vergleich mit der Adresse ADD 2. Mit der Entscheidung wird der Speicherplatz bestimmt. In das Flußdiagramm läßt sich eine weitere Möglichkeit einbauen. Was ist, wenn die beiden Adressen gleich sind. Hier haben wir nun die Möglichkeit, die ADD 1 auf SP 5 abzuspeichern.

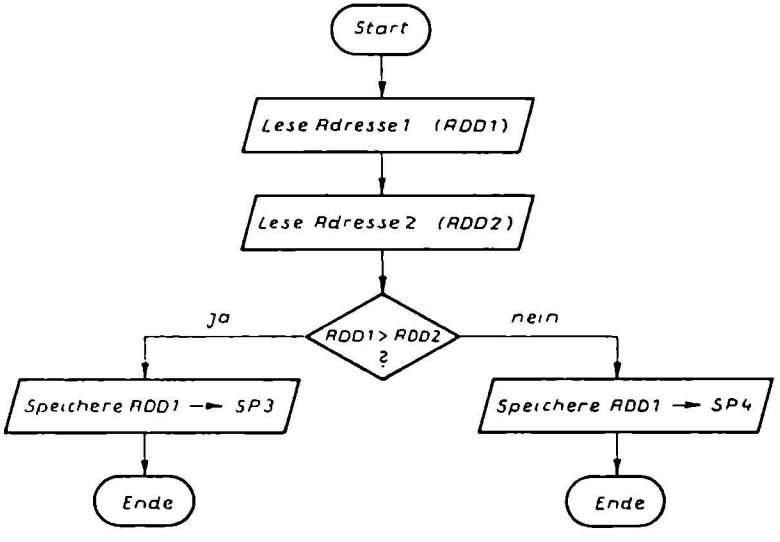


Bild 2.3.3 Flußdiagramm für einen Vergleicher

2.4. Einadreßmaschine

Mit der Einadreßmaschine können Flußdiagramme verarbeitet werden. Hierzu arbeitet die Maschine nach einer Instruktion oder dem Befehlswort. Jede Instruktion besteht aus zwei Teilen:
a) dem Adreßteil: In der Adresse befindet sich die Nummer des Speicherplatzes und in dem Speicherplatz ist die nächste Information für die Maschine eingespeichert.
b) dem Operationsteil: Dieser Teil enthält den Befehl, was in der Maschine mit der eingeholten Information geschehen soll.

Arbeitet man mit einem Byte, so ist eine Instruktion folgendermaßen aufgebaut:

Adreß-Teil	Operations-Teil
4 Bit	4 Bit

Jeder Teil einer Instruktion hat 4 Bits. Mit einer Einadreßmaschine können in dem Adreß-Teil 16 Adressen gespeichert werden, ebenso 16 Befehlswörter im Operations-Teil.
Programmiert man eine Einadreßmaschine, so ist es wichtig, daß die Instruktionen in fortlaufenden Speicherplätzen stehen. Bei den Mikroprozessoren ist dies nicht unbedingt erforderlich, aber es erhöht die Rechengeschwindigkeit.
In den nachfolgenden Kapiteln sind einfache Rechenbeispiele für eine Einadreßmaschine gezeigt. Mit den nachfolgenden Instruktionen im Operationsteil können folgende Befehle durchgeführt werden:

Befehl	uncodiert	codiert
Ende	0	L L L L
addiere	1	L L L H
Carry-Bit	2	L L H L
subtrahiere	3	L L H H
komplementiere	4	L H L L
speichere Akku	5	L H L H
korrigiere mit H	6	L H H L
korrigiere mit 6	7	L H H H
in das IR	8	H L L L
SZ + 1	9	H L L H
vergleiche SZ mit IR	A (10)	H L H L
springe	B (11)	H L H H
Akku > IR	C (12)	H H L L
speichere SZ	D (13)	H H L H
IT?	E (14)	H H H L
Fehler	F (15)	H H H H

Befehl 0 (LLLL): Mit dem Befehl 0 wird der Rechenvorgang beendet.
Befehl 1 (LLLH): Mit dem Befehl kann der Rechner addieren. Der Inhalt der Adresse wird mit dem Inhalt des Akkumulators (Akku) addiert. Beim 1. Rechenschritt ist der Akku leer, d.h. der Inhalt der Adresse wird mit 0 addiert.
Befehl 2 (LLHL): Der Befehl lautet: „Carry-Bit? ". Ein Carry-Bit entsteht, wenn wir im B-Komplement rechnen, wie dies im Kapitel 2.6 gezeigt ist. Hat ein Carry-Bit ein H-Signal, so wird ein Unterprogramm abgerufen. In dem Unterprogramm steht: „Akku + H". Der Inhalt des Akkus wird mit einem H-Signal nochmals mit einer 1 addiert, damit ist das Zwischenergebnis korrigiert und wir haben das Endergebnis.
Befehl 3 (LLHH): Den Befehl „subtrahiere" verwenden wir bei der Subtraktion. Von dem Inhalt des Akkumulators wird der Inhalt der Adresse subtrahiert. Bevor aber dieser Befehl ausgeführt werden kann, muß der Inhalt des Akkus bereits einen Wert angenommen haben.
Befehl 4 (LHLL): Mit dem Befehl „komplementiere" können wir den Inhalt einer Adresse komplementieren, d.h. umdrehen. Aus einem L wird ein H und aus H wird L.

L H L L
↓ ↓ ↓ ↓ } B-Komplement
H L H H

Das B-Komplement finden wir nur bei der indirekten Subtraktion.
Befehl 5 (LHLH): Durch den Befehl „speichere" können wir den Inhalt des Akkus abspeichern. Mit der Adresse legen wir den Speicherplatz fest, z.B. speichere den Inhalt des Akkus in Speicherplatz SP 14. In dem Speicherplatz SP 14 steht dann unser Ergebnis.

Befehl 6 (LHHL): Nach dem Befehl 2 wird dieses Unterprogramm abgerufen. Mit dem Befehl 6 können wir den Inhalt des Akkus mit einer 1 (H-Signal) korrigieren. Dieses Unterprogramm wird bei Einadreßmaschinen häufig als Befehl ausgeführt, um noch andere Rechenschritte zu programmieren.

Befehl 7 (LHHH): Tritt eine Pseudotetrade auf, so wird der Inhalt des Akkus mit 6 (LHHL) korrigiert. Dieser Befehl stellt ein Unterprogramm dar. Der Befehlsaufruf zum Korrigieren erfolgt meistens nach dem Befehl F, wenn eine Pseudotetrade PT festgestellt worden ist. Der Befehl 7 wird wie Befehl 6 als Befehlsunterprogramm geführt, um damit weitere Rechenschritte ausführen zu können.

Befehl 8 (HLLL): Mit dem Befehl „in das IR" können wir jeden Inhalt einer Adresse in das Index-Register geben.

Befehl 9 (HLLH): Dieser Befehl „SZ + 1" ist ein Unterprogramm. Bei jedem Additions- und Subtraktions-Umlauf notiert der Schleifen-Zähler zu seinem Inhalt eine 1 hinzu.

Befehl A (HLHL): Bei dem Befehl „vergleiche SZ mit IR" wird der Zählerstand des Schleifen-Zählers mit dem Inhalt des Index-Registers verglichen. Dieser Befehl kommt fast immer nach dem Befehl 9.

Befehl B (HLHH): Der Befehl „springe" stellt einen bedingten Sprung, also ein Unterprogramm dar. Mit der Adresse kann bestimmt werden, wohin der Sprung erfolgen soll. Ein bedingter Sprung erfolgt immer nur nach einem Vergleich oder einer Entscheidung. Durch den bedingten Sprung können wir eine Additions- oder Subtraktionsschleife aufbauen. Bei der Additionsschleife führen wir eine Multiplikation und bei der Subtraktionsschleife eine Division durch.

Befehl C (HHLL): Mit dem Befehl „Akku > IR" führen wir einen Vergleich zwischen dem Akkumulator und dem Index-Register durch. Diesen Befehl setzen wir nur bei der Division ein. Ist der Akku größer als das Index-Register, so wird auf ein Unterprogramm geschaltet, daß in der Regel einen bedingten Sprung, also einen Befehl B (HLHH) darstellt. Ist der Akku gleich oder kleiner als das Index-Register, so können wir einen Speicher-Befehl programmieren.

Befehl D (HHLH): Der Befehl „speichere SZ" speichert den Inhalt des Schleifen-Zählers in eine bestimmte Adresse. Diese Adresse wird durch den Adreß-Teil festgelegt.

Befehl E (HHHL): Mit dem Befehl „PT? " führen wir einen Vergleich aus. Ist keine Pseudotetrade vorhanden, dann erfolgt keine Korrektur des Akkumulators. Bei einer Pseudotetrade wird der Inhalt des Akkumulators mit 6 (LHHL) korrigiert. Dieser Fall tritt ein, wenn wir im BCD-Code arbeiten, d.h. wenn die Ausgabe im Dezimalzahlensystem erfolgt.

Befehl F (HHHH): Bei diesem Befehl „Fehler" bleibt die Einadreßmaschine stehen und zeigt einen Fehler an. Dieser Fehler liegt dann meistens in der Programmierung vor, wenn z.B. $\sqrt{-4}$ eingegeben wird.

2.5. Addition

Bei der Addition können die einfachen Rechenregeln angewendet werden:

3	≙	L L H H
4	≙	L H L L
7	≙	L H H H

Für die Einadreßmaschine soll ein Flußdiagramm erstellt werden:
Speicherplatzbelegung: Programm auf Plätze 0 bis 3
Daten auf Platz 4 (Ziffer 3)
auf Platz 5 (Ziffer 4)
auf Platz 6 (Ziffer 7)

In Bild 2.5.1 ist das Flußdiagramm für den Ablauf bei einer Addition in einer Einadreßmaschine gezeigt. Der Inhalt des Akkumulators (Akku) ist leer. Die 1. Instruktion lautet, „Inhalt des Akkus + SP 4". Der Inhalt des Speicherplatzes SP 4 wird in den Akku addiert. Nach der Ausführung der 1. Instruktion heißt der nächste Befehl „Akku + SP 5". Der Inhalt des Akkus wird nun mit dem Inhalt des Speicherplatzes SP 5 addiert. Die 3. Instruktion lautet „Akku nach SP 6". Der Inhalt, das Ergebnis der Addition, wird in den Speicherplatz SP 6 gegeben. In dem Speicherplatz SP 6 steht das Ergebnis.

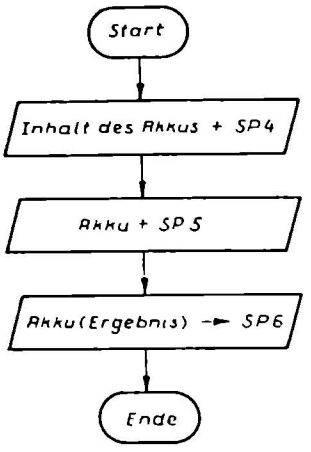


Bild 2.5.1 Flußdiagramm für eine Addition

SP4 = 3 (1. Summand)
SP5 = 4 (2. Summand)
SP6 = 7 (Summe, Ergebnis)

Programm für die Addition:

	Instruktion			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	4	1 (addiere)	L H L L	L L L H
1	5	1 (addiere)	L H L H	L L L H
2	6	5 (speichere)	L H H L	L H L H
3	–	0 (Ende)	L L L L	L L L L

In dem Speicherplatz 0 steht die 1. Instruktion. Diese lautet: „addiere SP 4 in Akku". Im Adreß-Teil steht, wo sich die Information, der Summand befindet. Im Operationsteil steht, was mit der Information geschehen soll. Mit dem Befehl 1 soll der Inhalt des Speicherplatzes 4 in den Akku addiert werden. Beim Mikroprozessor 8080 ist dies ein Transferbefehl mit dem Wort „MOV" oder „LHDDDSSS".
In dem Speicherplatz 1 steht die 2. Instruktion. Diese lautet: „addiere SP 5 in Akku". Da es sich um einen Addier-Befehl handelt, so liegt ein Befehl 1 im Operationsteil vor.
Der Speicherplatz 2 beinhaltet die 3. Instruktion. Der Befehl 5 im Operationsteil lautet aufgeschlüsselt: „Inhalt des Akkus ist in SP 6 zu speichern".
Auf dem Speicherplatz 3 wird das Ende des Flußdiagrammes programmiert. Eine 0 im Operations-Teil bedeutet: „Die Rechenoperation ist beendet".
Der Operations-Teil ist von System zu System unterschiedlich.

Die Addition im Dual-Code ist einfach, da nicht korrigiert wird, wie dies im BCD-Code der Fall ist.

Addition im Dual-Code

```
  6  ≙   L H H L
+ 7  ≙   L H H H
-------------------
 13  ≙   H H L H
```

Addition im BCD-Code

```
  6  ≙   L H H L
+ 7  ≙   L H H H
-------------------
 13  ≙   H H L H
         L H H L
-------------------
L L L H   L L H H
  1         3
```

im BCD-Code eine PT korrigiere mit 6

Wenn wir im BCD-Code die beiden Zahlen 6 und 7 addieren, so muß mit einer dualen 6 (LHHL) korrigiert werden, da HHLH eine Pseudotetrade ist. Mit dem Flußdiagramm von Bild 2.5.2 kann das Programm aufgestellt werden.
Programm für die Addition mit Korrektur:

	Instruktion			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	8	1 (addiere)	H L L L	L L L H
1	9	1 (addiere)	H L L H	L L L H
2	–	E (PT?)	L L L L	H H H L
3	–	7 (korrigiere mit 6)	L L L L	L H H H
4	10	5 (speichere)	H L H L	L H L H
5	–	0 (Ende)	L L L L	L L L L

In dem Speicherplatz 0 steht der Befehl 1 „addiere" den Adreß-Teil in den Akkumulator. Im Speicherplatz steht der nächste Addier-Befehl mit dem dazugehörigen Adreß-Teil. Im Speicherplatz 3 ist der Befehl 2 eingespeichert, der das Ergebnis der Addition mit einer Pseudotetradenerkennung vergleicht. Befindet sich das Ergebnis nicht in einer Pseudotetrade, so wird das Ergebnis der Addition in den Spei-

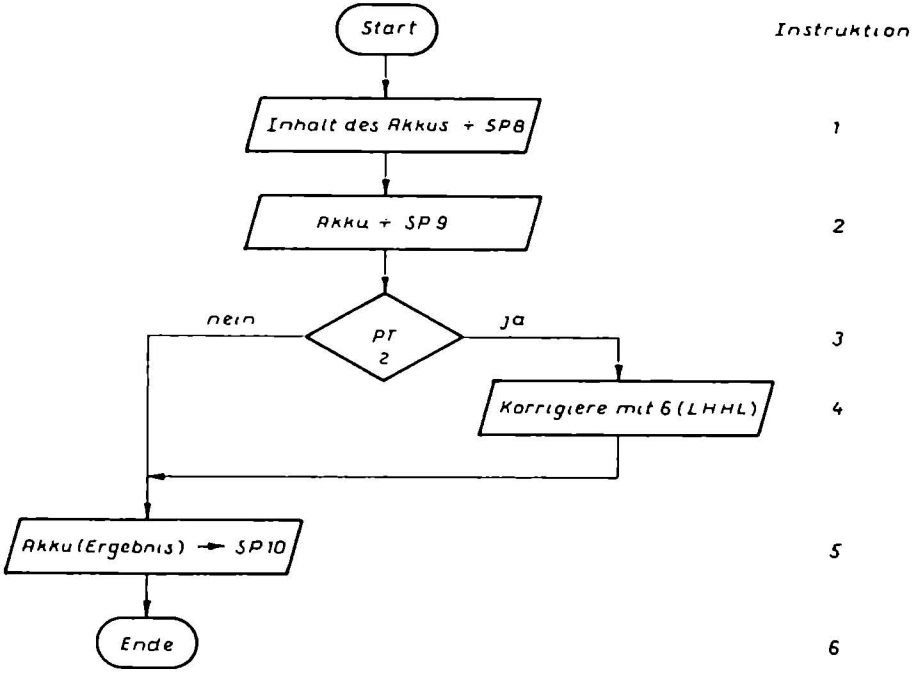


Bild 2.5.2 Flußdiagramm für eine Addition im BCD-Code

SP8 = 3 (1. Summand)
SP9 = 4 (2. Summand)
SP10 = 7 (Summe, Ergebnis)

cherplatz SP 10 gespeichert. Erkennt hingegen der Vergleicher eine Pseudotetrade, so tritt das Unterprogramm „korrigiere mit 6" ein. Nach der erfolgten Korrektur wird das Ergebnis in den Speicherplatz SP 10 gespeichert.
In dem Bild 2.5.2 ist ein Programmsprung eingezeichnet. Erkennt der Vergleicher keine Pseudotetrade, so springt das Programm von dem Befehl 3 auf den Befehl 5. Der Befehl 4 ist als Unterprogramm zu bewerten. Das Unterprogramm stellt einen bedingten Befehl dar, da es nur zur Korrektur verwendet werden muß. Der Sprung von Befehl 3 nach 5 stellt dagegen einen unbedingten Sprung dar.

2.6. Subtraktion

Die Subtraktion kann nach zwei Verfahren durchgeführt werden:
a) Direkte Subtraktion

```
  8 ≙ H L L L
- 4 ≙ L H L L
-------------
  4 ≙ L H L L
```

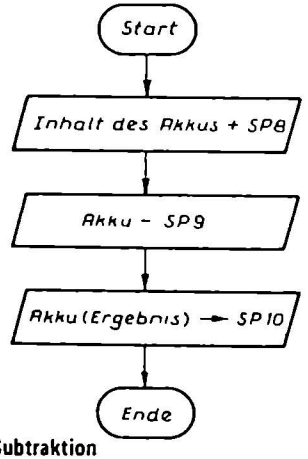
Die direkte Subtraktion erfolgt in einem Subtrahier-Rechenwerk.

b) Indirekte Subtraktion (B-Komplement)

```
8 ≙ H L L L  ⇒  bleibt     ⇒     H L L L
4 ≙ L H L L  ⇒  B-Kompl.   ⇒   + H L H H
                                ---------
                               H L L H H
                               └──────► H
                               ---------
                                 L H L L
```

Bei dem Verfahren mit dem B-Komplement wird der Minuend (8) direkt übernommen. Der Subtrahend hingegen wird komplementiert, d.h. er wird umgedreht. Aus einem L-Signal wird H und aus H wird L. Danach erfolgt eine Addition. Durch die Addition entsteht ein Überlauf, das sogenannte Carry-Bit. Dieses Carry-Bit muß zum ersten Additions-Ergebnis noch hinzuaddiert werden. Durch die zweite Addition erhalten wir das richtige Ergebnis.

Im Bild 2.6.1 ist der Ablauf einer direkten Subtraktion gezeigt.



Instruktion
1
2
3
4

Bild 2.6.1 Flußdiagramm für eine direkte Subtraktion

SP8 = 8 (Minuend)
SP9 = 4 (Subtrahend)
SP10 = 4 (Differenz)

Programm für die direkte Subtraktion:

	Instruktion			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	8	1 (addiere)	H L L L	L L L H
1	9	3 (subtrahiere)	H L L H	L L H H
2	10	5 (speichere)	H L H L	L H L H
3	–	0 (Ende)	L L L L	L L L L

Im Bild 2.6.2 ist der Ablauf einer indirekten Subtraktion gezeigt.

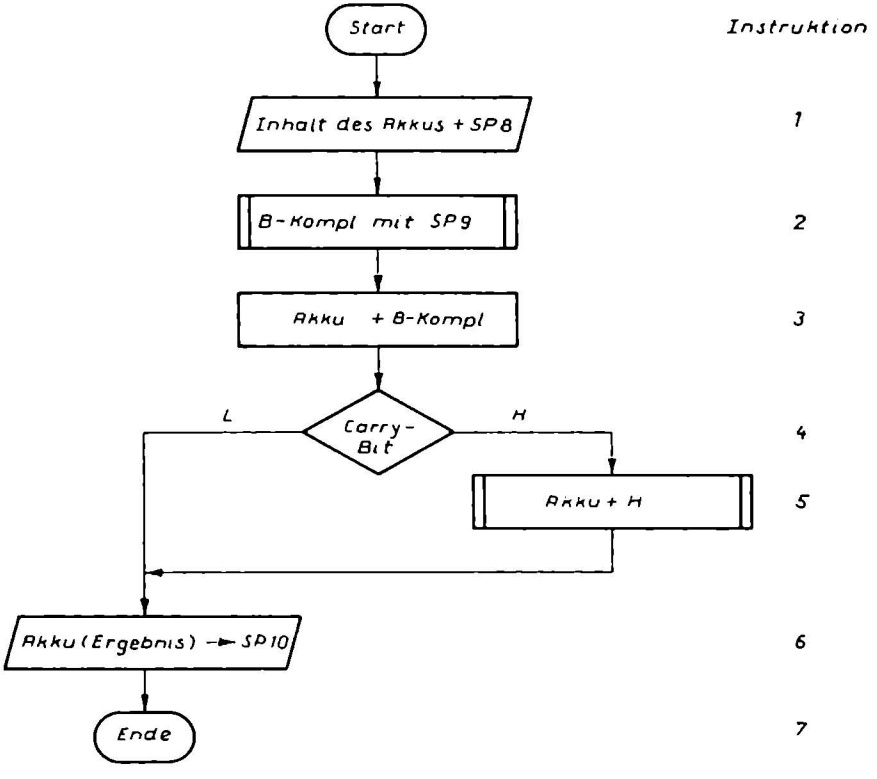


Bild 2.6.2 Flußdiagramm für eine indirekte Subtraktion

Programm für die indirekte Subtraktion:

	Instruktion			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	8	1 (addiere)	H L L L	L L L H
1	9	4 (Komplement)	H L L H	L H L L
2	–	1 (addiere)	L L L L	L L L H
3	–	2 (Carry-Bit?)	L L L L	L L H L
4	–	6 (korrigiere mit H)	L L L L	L H H L
5	10	5 (speichere)	H L H L	L H L H
6	–	0 (Ende)	L L L L	L L L L

2.7. Multiplikation

Die Multiplikation kann auf eine fortlaufende Addition zurückgeführt werden.

$$15 \cdot 4 = 60$$

Die 15 ist der Multiplikand und die 4 ist der Multiplikator. Die 60 ist das Ergebnis oder das Produkt.

$$15 + 15 + 15 + 15 = 15 \cdot 4$$

oder

```
  00
+ 15      1. Addition
----
  15
+ 15      2. Addition
----
  30
+ 15      3. Addition
----
  45
+ 15      4. Addition
----
  60
```

Nach der 4. Addition ist die fortlaufende Addition beendet, da der Multiplikator 4 ist.

In Bild 2.7.1 ist das Flußdiagramm für eine Multiplikation gezeigt.
Im Speicherplatz SP 8 steht der Multiplikator. Der Multiplikator wird in das Index-Register IR gespeichert. Bei den Mikroprozessoren kann jeder freie Speicherplatz als Index-Register verwendet werden. Danach erfolgt eine Addition des Akkus mit dem Speicherplatz SP 9, indem der Multiplikand steht. Mit dem Unterprogramm Schleifen-Zähler, „SZ + 1" wird jeder Additionsumlauf notiert. In dem Vergleicher wird der Schleifen-Zähler SZ mit dem Index-Register IR verglichen. Ist der Schleifen-Zähler SZ mit dem Index-Register IR ungleich, erfolgt durch die Addierschleife ein Sprung auf die Adresse 2. Ist der Schleifen-Zähler mit dem Index-Register gleich, so speichert die Einadreßmaschine das Produkt (Ergebnis) ab.

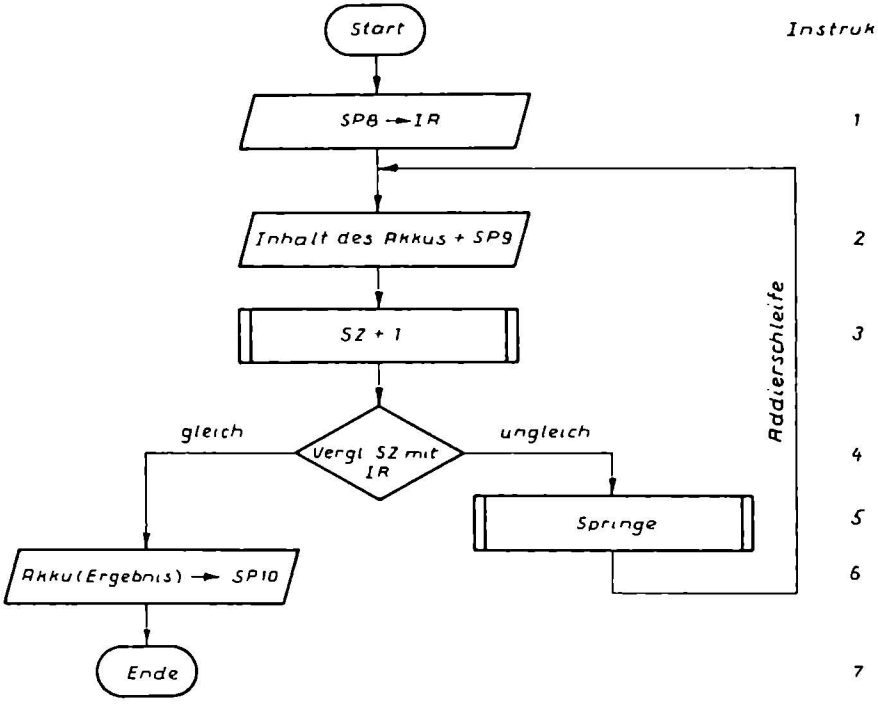


Bild 2.7.1 Flußdiagramm für eine Multiplikation

Programmierung für die Multiplikation:

	Instruktionen			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	8	8 (in das IR)	H L L L	H L L L
1	9	1 (addiere)	H L L H	L L L H
2	–	9 (SZ + 1)	L L L L	H L L H
3	–	A (vergleiche) SZ mit IR)	L L L L	L L H L
4	2	B (springe)	L L H L	H H L L
5	10	5 (speichere)	H L H L	L H L H
6	–	0 (Ende)	L L L L	L L L L

2.8. Division

Die Division kann auf eine fortlaufende Subtraktion zurückgeführt werden:

20 : 5 = 4

Die 20 ist der Dividend und die 5 der Divisor. Das Ergebnis 4 ist der Quotientwert.

```
  20
- 5        1. Subtraktion
  15
- 5        2. Subtraktion
  10
- 5        3. Subtraktion
   5
- 5        4. Subtraktion
   0
```

Nach der 4. Subtraktion ist die fortlaufende Subtraktion beendet.
In Bild 2.8.1 ist das Flußdiagramm für eine Division gezeigt. Im Speicherplatz SP 9 steht der Divisor. Der Divisor wird in das Index-Register IR gespeichert. Im Speicherplatz SP 10 steht der Dividend, der in den Akku addiert wird. Danach erfolgt Akku minus SP 9. Im Akku steht der Dividend von dem der Divisor subtrahiert werden soll. Der Schleifen-Zähler SZ notiert jeden Subtraktionsumlauf. Im Vergleicher wird der Inhalt des Akkus mit dem Index-Register verglichen. Ist der Akku-Inhalt kleiner als IR, so wird der Inhalt des Schleifen-Zählers in den Speicherplatz SP 11 gespeichert. Der Schleifenzähler stellt das Ergebnis, also den Quotientwert dar.

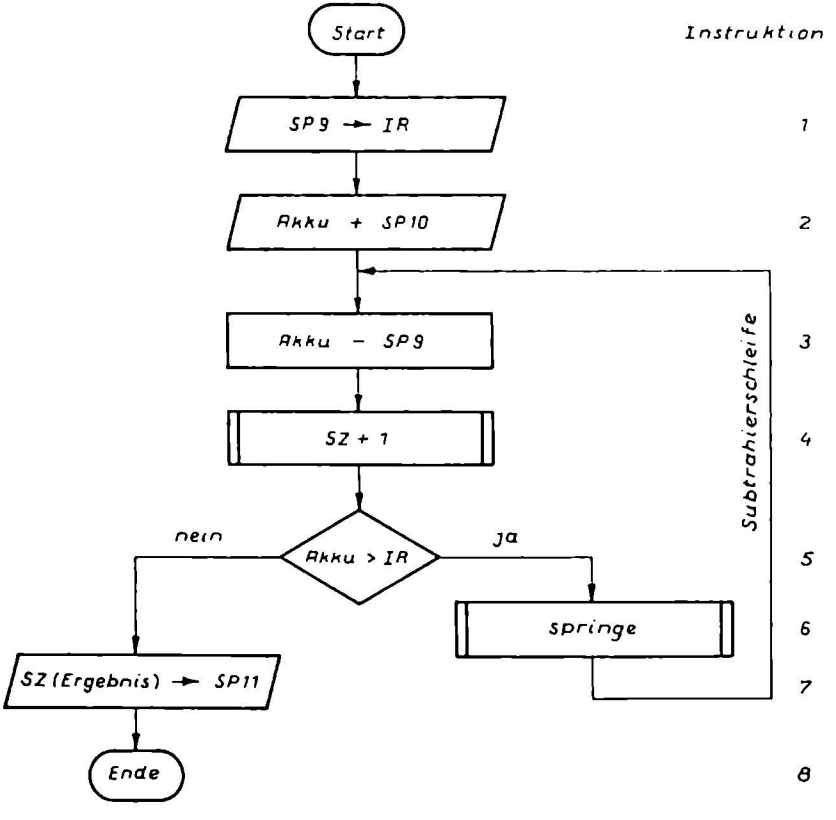


Bild 2.8.1 Flußdiagramm einer Division ohne Rest

Programmierung für eine Division ohne Rest:

	Instruktionen			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	9	8 (in das IR)	H L L H	H L L L
1	10	1 (addiere)	H L H L	L L L H
2	–	3 (subtrahiere)	L L L L	L L H H
3	–	9 (SZ + 1)	L L L L	H L L H
4	–	C (Akku > IR)	L L L L	H H L L
5	3	B (springe)	L L H H	H L H H
6	11	D (speichere SZ)	H L H H	H H L H
7	–	0 (Ende)	L L L L	L L L L

Divisionen ohne Rest treten in der Praxis nur sehr selten auf.

22 : 5 = 4 Rest 2

In Bild 2.8.2 ist das Flußdiagramm für eine Division mit Rest gezeigt. Das Diagramm ist im wesentlichen mit dem Bild 2.8.1 identisch, außer einem zusätzlichen Befehl.

22	
– 5	1. Subtraktion
17	
– 5	2. Subtraktion
12	
– 5	3. Subtraktion
7	
– 5	4. Subtraktion
2	

Nach der 4. Subtraktion ist der Inhalt des Akkus nicht mehr gleich oder größer wie das Index-Register. Die 7. Instruktion lautet „SZ nach SP 12". Die 8. Instruktion stellt den zusätzlichen Befehl dar „Akku nach SP 13". Auf dem Speicherplatz SP 12 steht so der Quotient und in Speicherplatz SP 13 der Rest.

Programmierung für eine Division mit Rest:

	Instruktionen			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	10	8 (in das IR)	H L H L	H L L L
1	11	1 (addiere)	H L H H	L L L H
2	–	3 (subtrahiere)	L L L L	L L H H
3	–	9 (SZ + 1)	L L L L	H L L H
4	–	C (Akku > IR)	L L L L	H H L L
5	3	B (springe)	L L H H	H L H H
6	12	D (speichere SZ)	H H L L	H H L H
7	13	5 (speichere Akku)	H H L H	L H L H
8	–	0 (Ende)	L L L L	L L L L

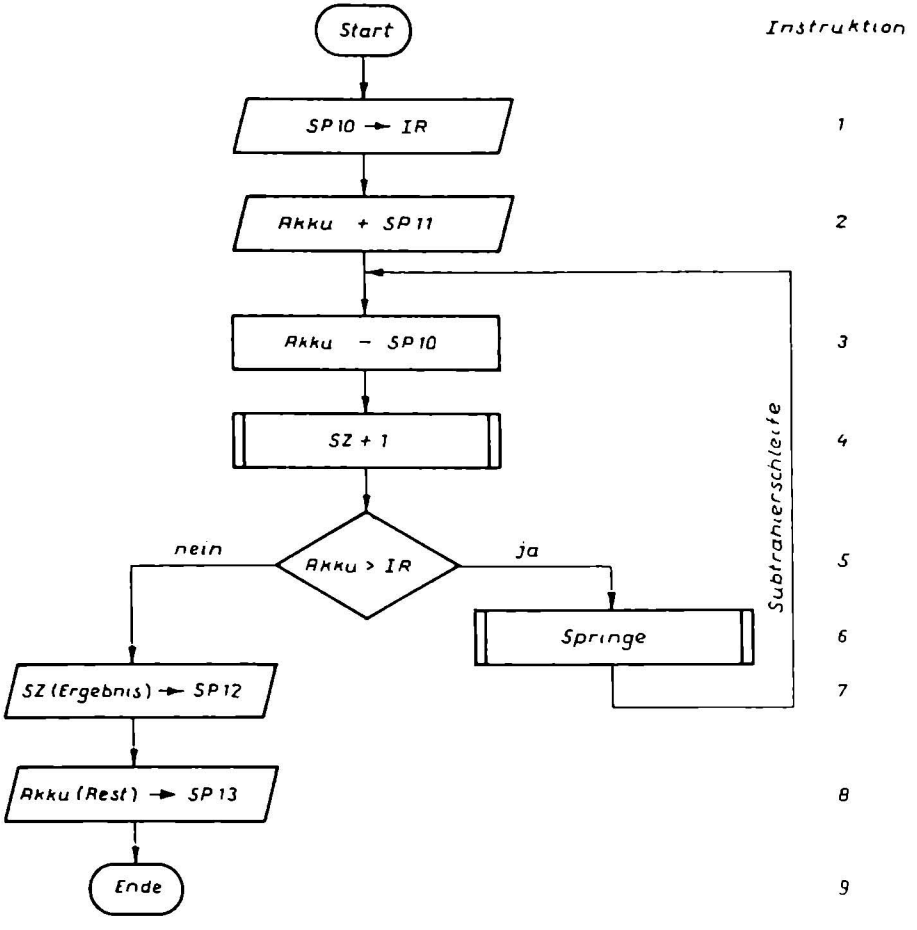


Bild 2.8.2 Flußdiagramm einer Division mit Rest

2.9. Quadrieren

Eine Zahl quadrieren heißt, wir müssen die Zahl mit sich selbst multiplizieren.

$3^2 = 9$ oder $3 \cdot 3 = 9$
$4^2 = 16$ oder $4 \cdot 4 = 16$

In Bild 2.9.1 ist ein Flußdiagramm für das Quadrieren von Zahlen gezeigt. Mit der 1. Instruktion wird der Inhalt des Speicherplatzes SP 8 in das Index-Register IR gegeben. Bei der 2. Instruktion erfolgt ein nochmaliges Auslesen der Information des SP 8 in den Akkumulator. Der Akku addiert den SP 8. In der 3. Instruktion zählt der Schleifen-Zähler SZ den Umlauf jeder Addition.

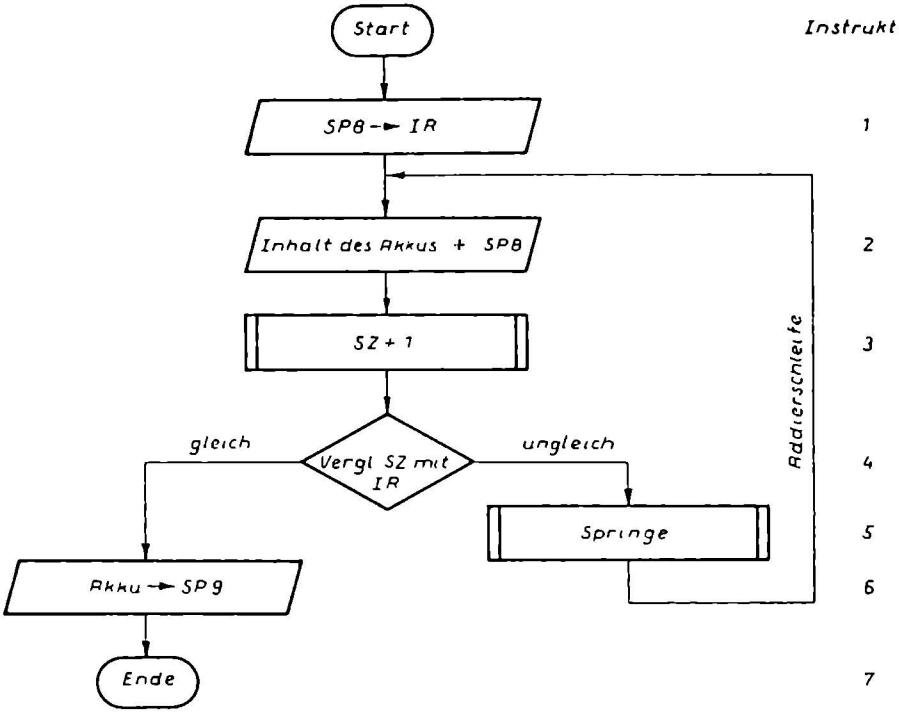


Bild 2.9.1 Flußdiagramm zum Quadrieren einer Zahl

Programmierung einer Quadrierung:

	Instruktionen			
	uncodiert		codiert	
Speicherplatz	Adreß-Teil	Operations-Teil	Adreß-Teil	Operations-Teil
0	8	8 (in das IR)	H L L L	H L L L
1	8	1 (addiere)	H L L L	L L L H
2	–	9 (SZ + 1)	L L L L	H L L H
3	–	A (vergleiche SZ mit IR)	L L L L	H L H L
4	–	B (springe)	L L L L	H L H H
5	9	5 (speichere Akku)	H L L H	L H L H
6	–	0 (Ende)	L L L L	L L L L

3. Mikrocomputer

Seit geraumer Zeit liefern die Halbleiter-Firmen komplette Mikrocomputer auf einem Chip, die sogenannten „Ein-Chip-Rechner".

3.1. Ein-Chip-Rechner

In Bild 3.1.1 ist ein Ein-Chip-Rechner mit Eingabe-Tastatur, Anzeige-Einheit und den dazugehörigen Treiber-Chips gezeigt.

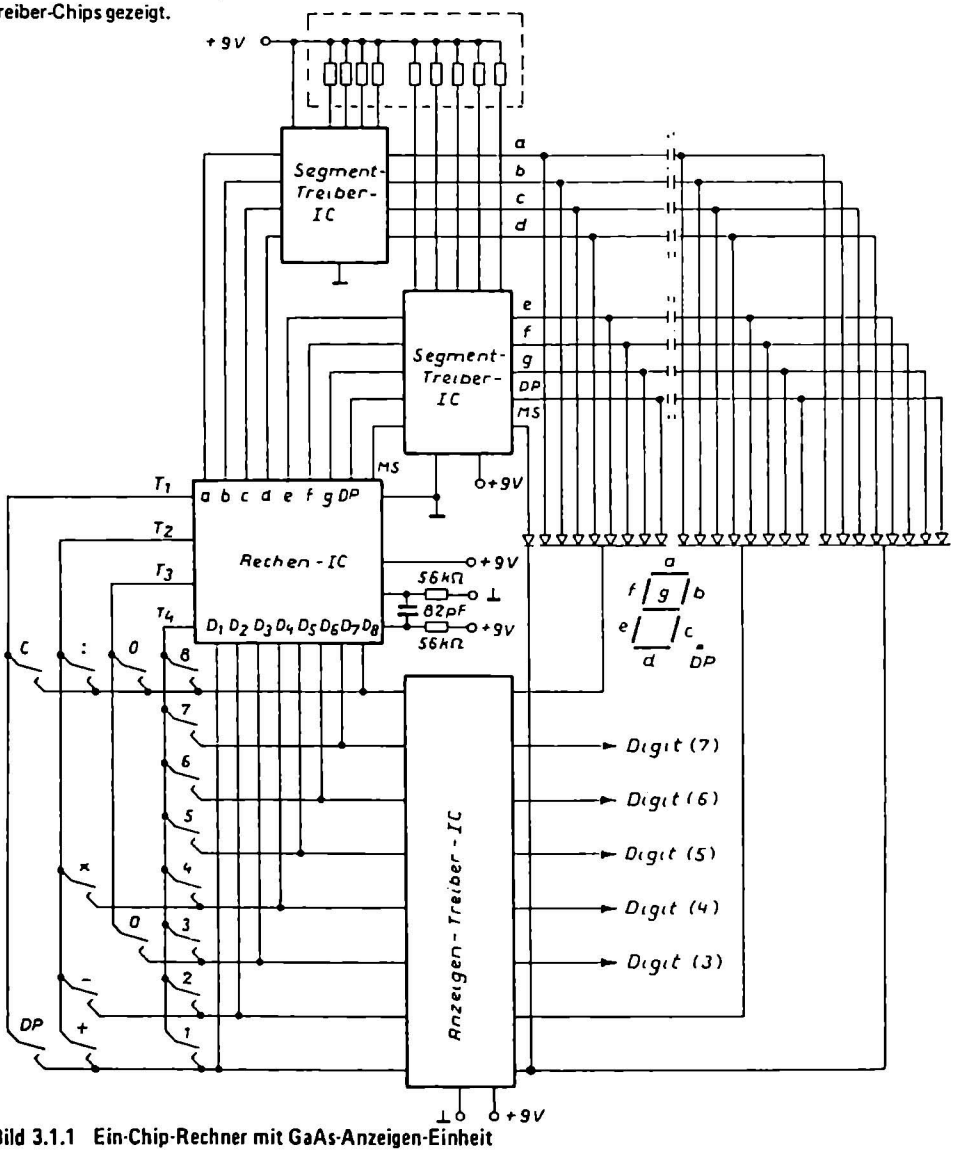


Bild 3.1.1 Ein-Chip-Rechner mit GaAs-Anzeigen-Einheit

Der Rechner befindet sich in einem 28-poligen DIL-Gehäuse. Die Betriebsspannung beträgt +9V. Die beiden Widerstände und der Kondensator erzeugen die Taktfrequenz für den Rechner, die Rechenfrequenz.

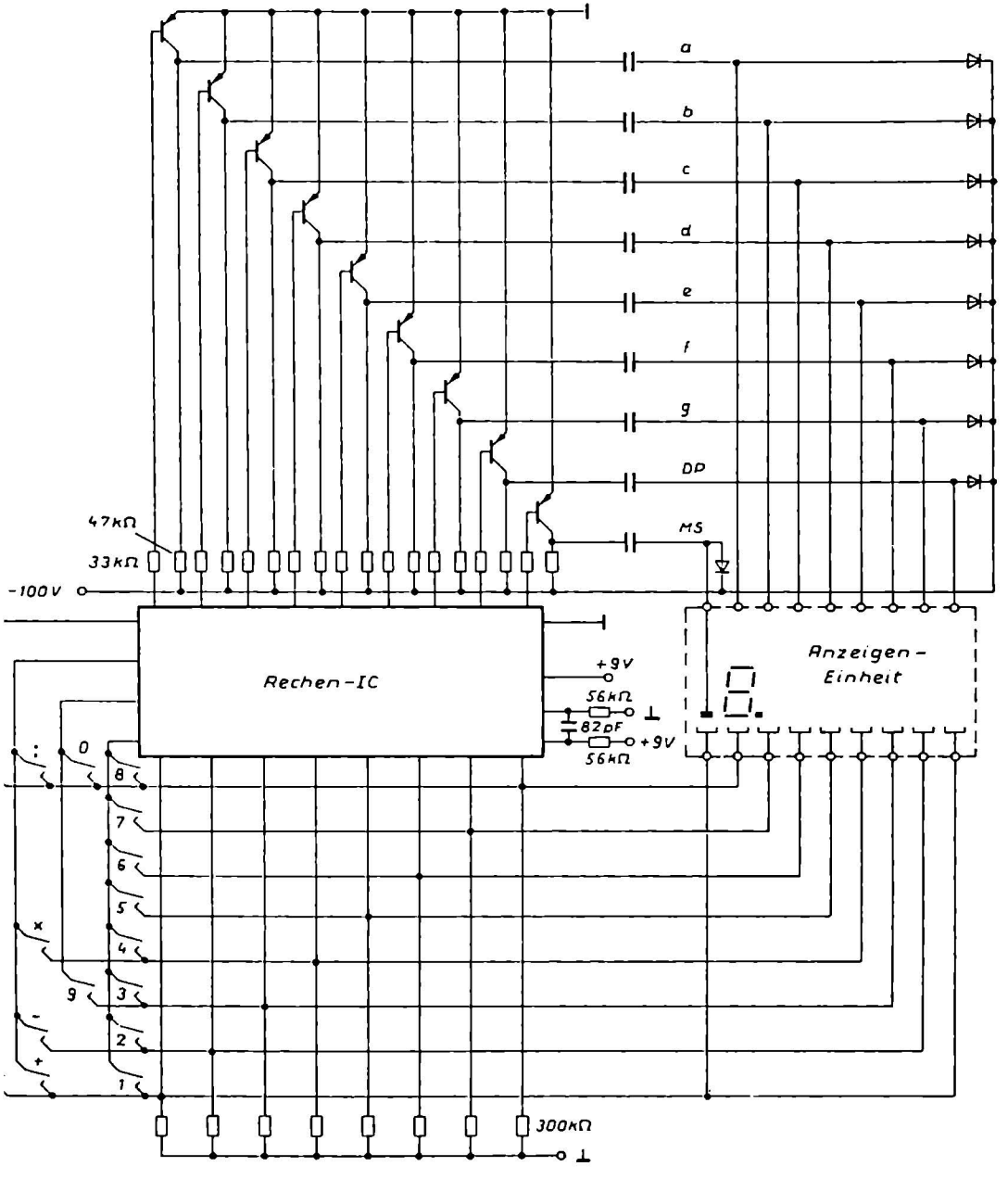


Bild 3.1.2 Ein-Chip-Rechner mit Neon-Anzeigen-Einheit

Die sieben Ausgänge a, b, c, d, e, f und g sind die Segment-Ausgänge. Sie liegen direkt an den Eingängen der Segment-Treiber-IC's an. Die Ausgänge der beiden Segment-Treiber-IC's sind sinngemäß mit den einzelnen Anoden der GaAs-Leuchtdioden verbunden. Jede a-Leuchtdiode ist mit dem a-Ausgang des IC's verbunden. Dasselbe gilt auch für die anderen Leuchtdioden. Insgesamt haben wir in Bild 3.1.1 acht 7-Segment-Anzeigen.
Der DP-Ausgang (Dezimal-Punkt) liegt ebenfalls über das Segment-Treiber-IC an allen Dezimal-Punkt-Leuchtdioden an.
Der MS-Ausgang (Minus-Zeichen) bildet eine Ausnahme. Vor der achten Stelle der Anzeigeneinheit liegt eine Leuchtdiode, die uns anzeigt, ob das Ergebnis negativ ist.
Die acht D-Ausgänge sind die Digit-Ausgänge. Sie steuern das Anzeigen-Treiber-IC und außerdem die Eingabe-Tastatur an. Mit den Anzeigen-Treiber-IC's werden die Katoden der 7-Segment-Anzeigen angesteuert. Die Ausgänge dieses IC's steuern nacheinander jede Anzeigen-Einheit an. Die Anzeige wird also im Multiplex-Verfahren betrieben.
Über die T-Eingänge erfolgt die Eingabe der Zahlen und Rechenbefehle in den Rechner.

In Bild 3.1.2 ist ein Rechner mit einer Neon-Anzeige gezeigt. Bei dieser Anzeigen-Einheit werden keine Treiber-IC's benötigt, da die Stromaufnahme der Neon-Segmente sehr gering ist. Die Transistoren dienen zur Entkopplung der 100V Betriebsspannung.

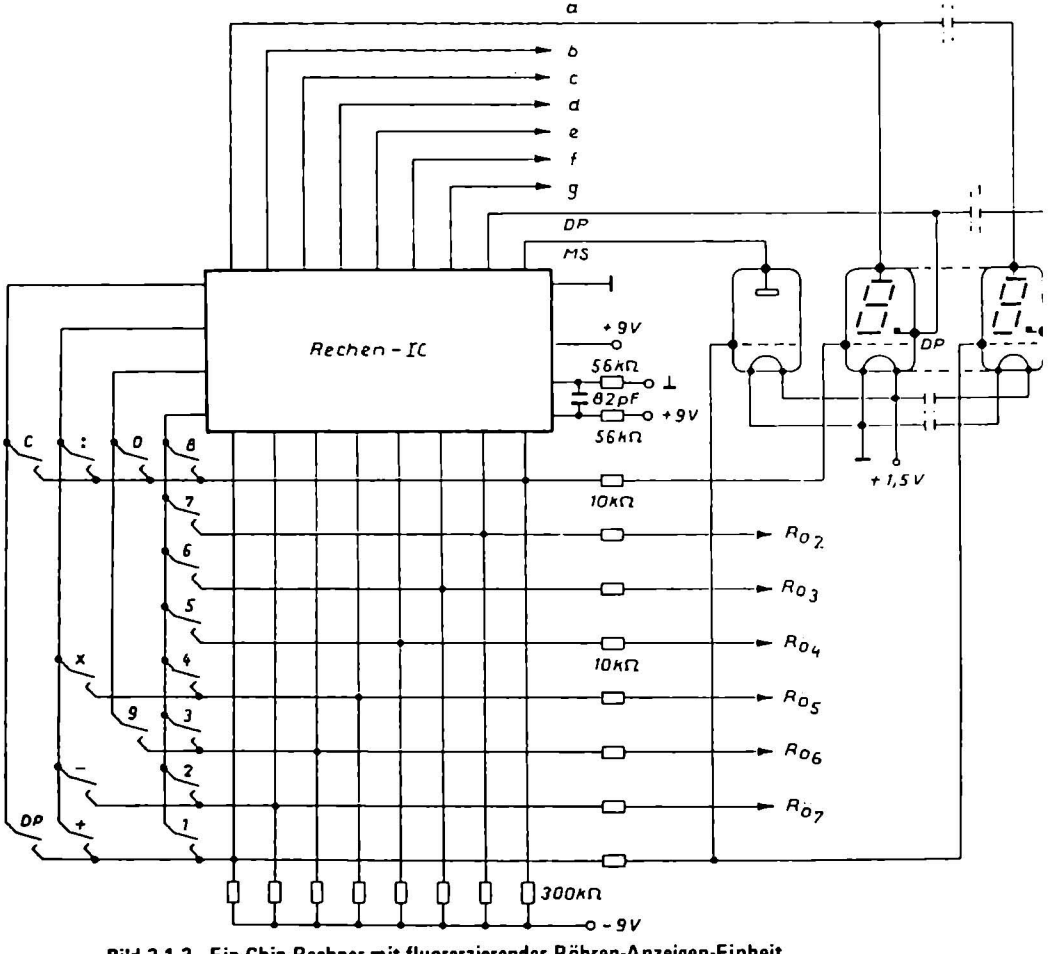


Bild 3.1.3 Ein-Chip-Rechner mit fluoreszierender Röhren-Anzeigen-Einheit

In Bild 3.1.3 ist ein Rechner mit einer grünleuchtenden Anzeige gezeigt. Bei dieser Anzeige sind keine Treiber-IC's und keine Transistoren erforderlich. Als Heizspannung für die fluoreszierenden Röhren dient eine 1,5V Batterie.

3.2. Innenschaltung eines Ein-Chip-Rechners

In Bild 3.2.1 ist das Blockschaltbild für die Innenschaltung eines Ein-Chip-Rechners gezeigt. Insgesamt besteht ein Ein-Chip-Rechner aus sieben Funktionsblöcken:

a) Anzeigen-Ansteuerung
b) Segment-Ansteuerung
c) Eingabe-Logik
d) Eingangs-Logik
e) Instruktions- und Kontrollregister
f) Anzeigen-Register und Akkumulator
g) Arithmetik-Einheit

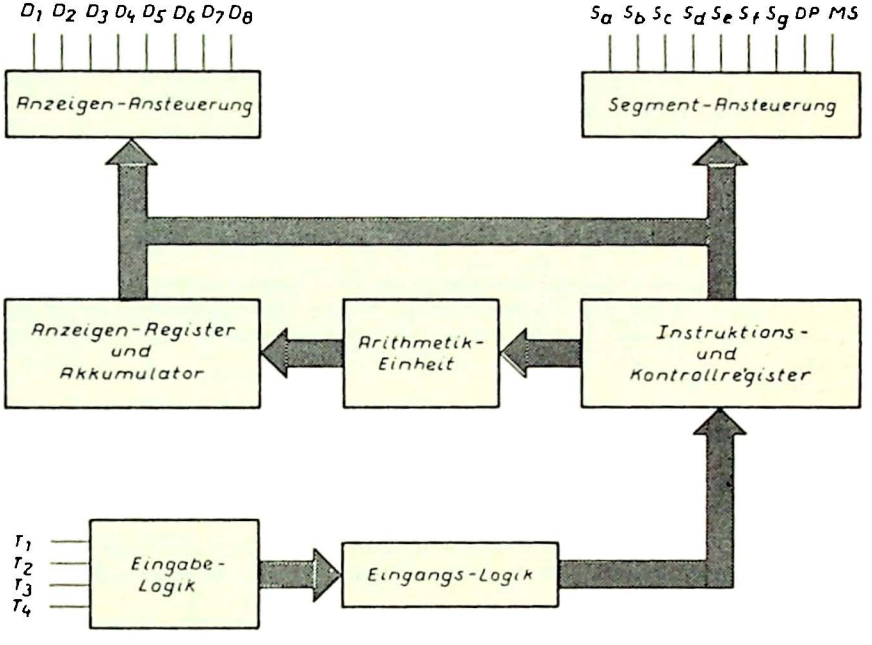


Bild 3.2.1 Blockschaltbild für einen Ein-Chip-Rechner

Die Ausgabe der Informationen, also die eingegebenen Werte und die einzelnen Ergebnisse, erfolgen über eine achtstellige 7-Segment-Anzeige. In Bild 3.2.2 ist die Anzeige gezeigt. Es wird in diesem Bild folgender Zahlenwert dargestellt: – 123,45678.

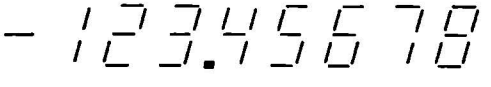


Bild 3.2.2 Achtstelliges Anzeigen-System eines Ein-Chip-Rechners mit Minus-Zeichen MS und Dezimalpunkt DP.

Der Dezimalpunkt DP in der Anzeige kann an jeder beliebigen Stelle aufleuchten. Wir sprechen hierbei von einer Gleitkomma-Darstellung. Dadurch können wir folgende Zahlenbeispiele darstellen:

1 0 2 4 8 1 9 3	ohne DP, also an der 8. Stelle
1 0 2 4 8 1 9,3	DP an der 7. Stelle
1 0 2 4 8 1,9 3	DP an der 6. Stelle
1 0 2 4 8,1 9 3	DP an der 5. Stelle
1 0 2 4,8 1 9 3	DP an der 4. Stelle
1 0 2,4 8 1 9 3	DP an der 3. Stelle
1 0,2 4 8 1 9 3	DP an der 2. Stelle
1,0 2 4 8 1 9 3	DP an der 1. Stelle

Bei diesen Beispielen wurden die acht Anzeigen benötigt. Die meisten Ein-Chip-Rechner neuerer Bauart sind mit einer Gleitkomma-Darstellung ebenso ausgestattet, wie mit einer automatischen Null-Unterdrückung. Die nachfolgenden Beispiele zeigen die Funktion der automatischen Null-Unterdrückung:

1 : 2	=	0,5
1 : 3	=	0,3 3 3 3 3 3 3
1 : 4	=	0,2 5
1 : 5	=	0,2
1 : 6	=	0,1 6 6 6 6 6 6
1 : 7	=	0,1 4 2 8 5 7 1
1 : 8	=	0,1 2 5
1 : 9	=	0,1 1 1 1 1 1 1
1 : 10	=	0,1
1 : 11	=	0,0 9 0 9 0 9 0

Die achtstellige 7-Segment-Anzeigen-Einheit wird im Multiplex-Verfahren angesteuert. In Bild 3.2.3 ist das Zeitdiagramm für den Multiplex-Betrieb gezeigt. Mit den Digit-Ausgängen D_1 bis D_8 werden die einzelnen Anzeigen angesteuert. Synchron zu den Digit-Impulsen liegen die Impulse der Segment-Ansteuerung.
Beispiel für Bild 3.2.3:

Der Digit-Ausgang D_1 hat ein L-Signal. Da eine 0 dargestellt werden soll, müssen alle Segment-Ausgänge bis auf S_g ein H-Signal aufweisen. Das Segment S_g bleibt dunkel, die anderen Segmente leuchten und es wird dadurch die Zahl hergestellt.

In Bild 3.2.3 sind alle Möglichkeiten der Zeichendarstellung noch nicht gezeigt. Viele Ein-Chip-Rechner zeigen ein E (error) an, wenn eine nicht lösbare Aufgabe in den Rechner eingegeben wird.

Die Eingabe der Daten erfolgt über eine Tastatur. Bei einfachen Rechnern besteht sie aus 16 Tasten, die nach Bild 3.2.4 angeordnet sind. Über die einzelnen Taster können wir die Daten in den Rechner eingeben.
Das Zeitdiagramm für die Daten-Eingabe ist in Bild 3.2.5 gezeigt. Mit den Digit-Ausgangssignalen wird auch die Tastatur angesteuert. Drücken wir beispielsweise die Taste „4", so erhält die Eingabe-Logik ein L-Signal.
Der Umlaufzyklus der Digit-Ausgänge liegt bei 8kHz, d.h. innerhalb von 0,125ms wird die eingegebene Information in die Eingabe-Logik übernommen. Die Abfrage der Tasten dauert 1ms, d.h. jede eingegebene Information wird nach maximal 1ms in die Eingabe-Logik übernommen.

Die Eingabe-Logik ist eine Demultiplex-Schaltung. Aus den parallel eingegebenen Informationen des Multiplex-Betriebes werden serielle Informationen erzeugt. Die Eingangs-Logik speichert die Informationen zwischen und gibt sie an das Instruktions- und Kontrollregister weiter.
Von dem Instruktionsregister wird die Arithmetik-Einheit, die ALU, angesteuert. In Bild 3.2.6 ist der Aufbau einer 4-Bit-Arithmetik-Logik-Einheit gezeigt.

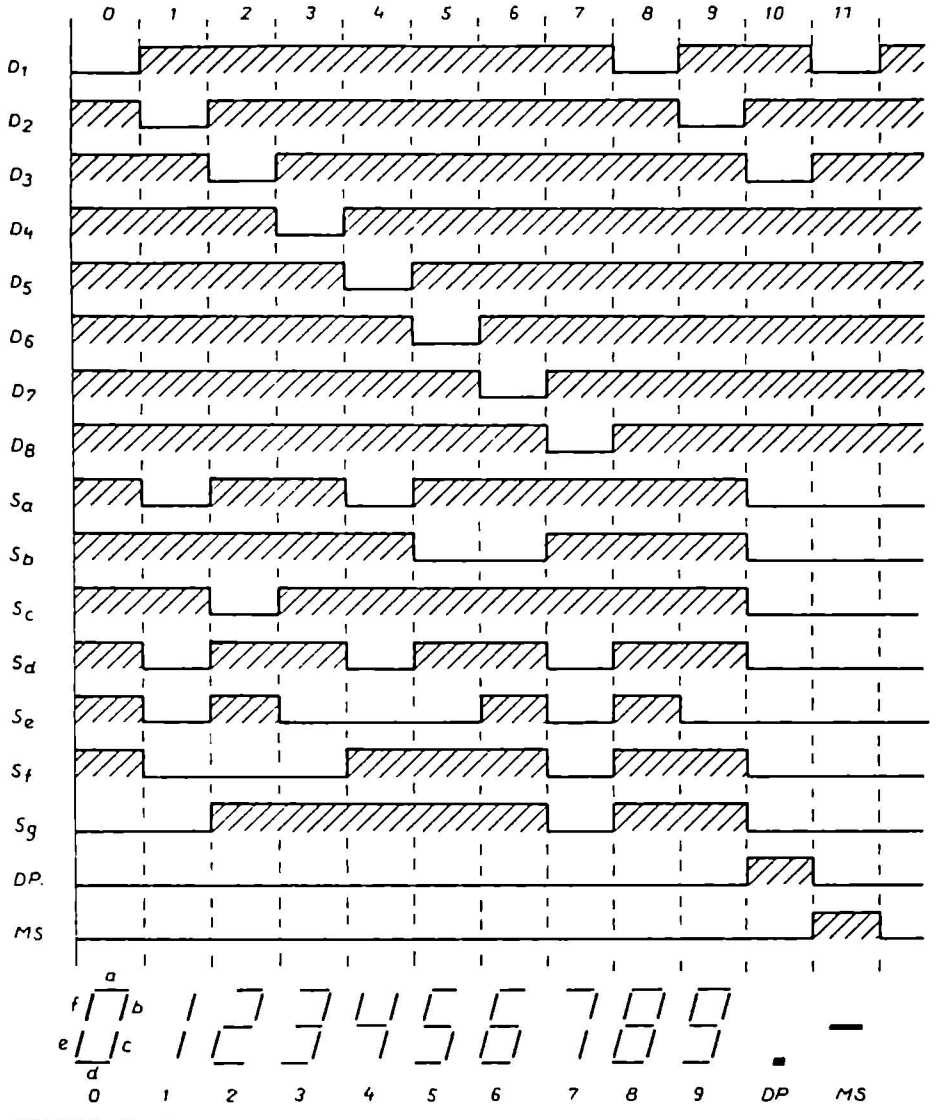


Bild 3.2.3 Zeitdiagramm für ein Anzeigen-System

D_1 bis D_8 = Digit-Ausgänge des Rechen-C's
S_a bis S_g = Leucht-Segmente der 7-Segment-Anzeigen

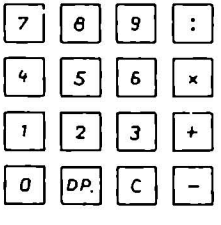


Bild 3.2.4 Tastatur eines Ein-Chip-Rechners

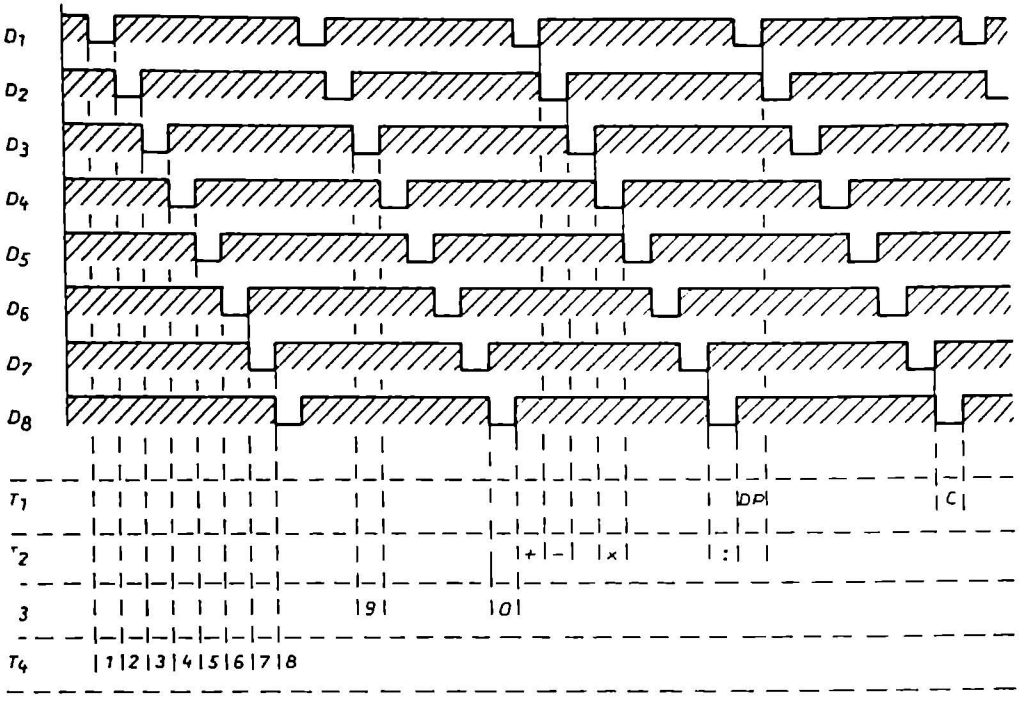


Bild 3.2.5 Zeitdiagramm für die Eingabe von Informationen über die Tastatur von Bild 3.2.4

Funktionsauswahl				Logik-Funktionen BA = H	Arithmetische Operationen BA = L	
					C_n = H (kein Übertrag)	C_n = L (Übertrag)
S_3	S_2	S_1	S_0			
L	L	L	L	$F = \bar{a}$	F = a	F = a plus 1
L	L	L	H	$F = \overline{a + b}$	F = a + b	F = (a + b) plus 1
L	L	H	L	$F = \bar{a} \cdot b$	F = a + $\bar{b}$	F = (a + $\bar{b}$) plus 1
L	L	H	H	F = L	F = H	F = L
L	H	L	L	$F = \overline{a \cdot b}$	F = a plus (a · $\bar{b}$)	F = a plus (a · $\bar{b}$) plus 1
L	H	L	H	F = $\bar{b}$	F = (a + b) plus (a · $\bar{b}$)	F = (a + b) plus (a · $\bar{b}$) plus 1
L	H	H	L	F = (a · $\bar{b}$) + ($\bar{a}$ · b)	F = a minus b minus 1	F = a minus b
L	H	H	H	F = a · $\bar{b}$	F = (a · $\bar{b}$) minus 1	F = a · $\bar{b}$
H	L	L	L	F = $\bar{a}$ + b	F = a plus (a · b)	F = a plus (a · b) plus 1
H	L	L	H	F = ($\bar{a}$ · $\bar{b}$) + (a · b)	F = a plus b	F = a plus b plus 1
H	L	H	L	F = b	F = (a + $\bar{b}$) plus (a · b)	F = (a + $\bar{b}$) plus (a · b) plus 1
H	L	H	H	F = a · b	F = (a · b) minus 1	F = a · b
H	H	L	L	F = H	F = a plus a*)	F = a plus a plus 1
H	H	L	H	F = a + $\bar{b}$	F = (a + b) plus a	F = (a + b) plus a plus 1
H	H	H	L	F = a + b	F = (a + $\bar{b}$) plus a	F = (a + $\bar{b}$) plus a plus 1
H	H	H	H	F = a	F = a minus 1	F = a

*) Jedes Bit wird zur höherwertigen Stelle geschoben

BA = Betriebsart

Die Schaltung in Bild 3.2.6 zeigt eine sehr schnelle arithmetische Logikeinheit, die 16 binäre arithmetische Operationen an zwei 4-Bit-Wörtern durchführt oder aber 16 logische Funktionen zweier Boolescher Variablen bildet.

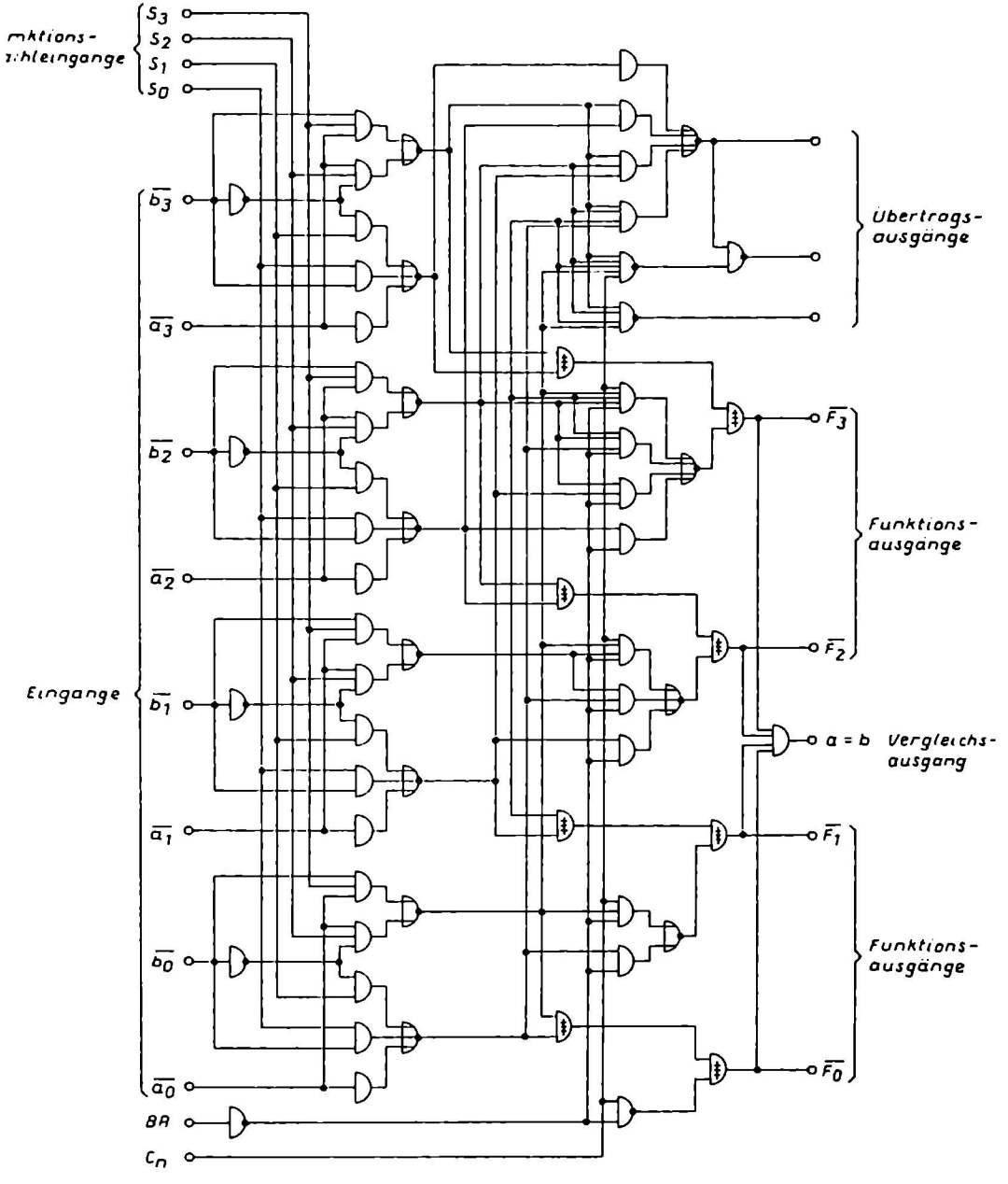


Bild 3.2.6 Aufbau einer 4-Bit-Arithmetik-Logik-Einheit

Bei der Funktionstabelle wurden daher zwei Abschnitte gewählt. Der linke Abschnitt ist für die Logik-Funktionen, wenn der BA-Eingang auf H-Signal liegt. Arithmetische Operationen können wir durchführen, wenn der BA-Eingang auf L-Signal liegt.

BA-Eingang auf H-Signal: Durch den Betriebsart-Eingang BA arbeitet die Schaltung von Bild 3.2.6 mit Logik-Funktionen. Mit den vier Funktionswahl-Eingängen S_0, S_1, S_2 und S_3 legen wir die Logik-Funktionen fest.

S_3	S_2	S_1	S_0	
L	L	L	L	$F = \bar{a}$, alle a-Eingänge werden invertiert
L	L	L	H	$F = \overline{a + b}$, alle a- und b-Eingänge werden mit NOR verknüpft
L	L	H	L	$F = \bar{a} \cdot b$, a invertiert und mit b zu UND verknüpft
L	L	H	H	F = L, Ausgänge haben ein L-Signal
L	H	L	L	$F = \overline{a \cdot b}$, a- und b-Eingänge über NAND verknüpft
L	H	L	H	$F = \bar{b}$, die b-Eingänge werden invertiert
L	H	H	L	$F = (a \cdot \bar{b}) + (\bar{a} \cdot b)$, a und b über Exklusiv-ODER verknüpft
L	H	H	H	$F = a \cdot \bar{b}$, a mit invertiertem b-Eingang über UND verknüpft
H	L	L	L	$F = \bar{a} + b$, der invertierte a-Eingang ist mit b über ODER verknüpft
H	L	L	H	$F = (\bar{a} \cdot \bar{b}) + (a \cdot b)$, a und b über Inklusiv-ODER verknüpft
H	L	H	L	F = b, an den Ausgängen liegt eine b-Eingangsinformation an
H	L	H	H	$F = a \cdot b$, die a- und b-Eingänge sind über UND verknüpft
H	H	L	L	F = H, sämtliche Ausgänge haben ein H-Signal
H	H	L	H	$F = a + \bar{b}$, a mit invertiertem b über ODER verknüpft
H	H	H	L	F = a + b, a mit b über ODER verknüpft
H	H	H	H	F = a, an den Ausgängen liegt die a-Information an

Durch die vier Funktionswahl-Eingänge ergeben sich 16 verschiedene Funktionen nach der Booleschen Algebra.

BA-Eingang auf L-Signal: Mit dem L-Signal arbeitet die Schaltung in Bild 3.2.6 als Rechenelement. Hierbei unterscheiden wir zwischen zwei Betriebsarten. Bei der ersten Betriebsart hat der Übertrags-Eingang C_n ein H-Signal und bei der anderen ein L-Signal.
Wir können Rechenoperationen wie die Addition und Subtraktion durchführen. Die Subtraktion kann nur mit dem B-1-Komplement durchgeführt werden. Das B-1-Komplement des Subtrahenden kann dabei intern erzeugt werden. Das Ergebnis ist dann a minus b minus 1 und erfordert einen positiven Eingangsübertrag oder eine End-Aufrundung.

Mit der Schaltung von Bild 3.2.7, zwei 4-Bit-Umkehrschieberegister und acht Tri-State-Treiber können wir ein 4-Bit-Parallel-Rechenwerk zum Addieren, Subtrahieren, Multiplizieren und Dividieren aufbauen. In Bild 3.2.8 ist ein Schieberegister für rechte und linke Schieberichtungen gezeigt. Das Schieberegister besteht aus vier Flip-Flops. Die Takt-Eingänge der vier Flip-Flops sind parallel geschaltet und über eine elektronische Weiche mit den drei Eingängen verbunden, S_L = Takteingang zum Linksschieben der Informationen, S_R = Takteingang zum Rechtsschieben der Informationen und BA = Betriebsart, ob links/rechts oder ob seriell/parallel die Eingangsinformationen übernommen werden sollen.

BA = L: Das Schieberegister schiebt die eingespeicherten Informationen von links nach rechts. Wir haben ein rechtsschiebendes Register. In dieser Betriebsart können keine Daten parallel eingegeben werden, da die E-Eingänge gesperrt sind. Wir können nur serielle Daten über den SE-Eingang eingeben.

BA = H: Das Schieberegister schiebt die eingespeicherten Informationen von rechts nach links. Wir haben damit ein linksschiebendes Register. In dieser Betriebsart können wir Daten seriell eingeben, d.h. Informationen, die an den E-Eingängen anliegen, werden mit einem Taktimpuls an dem S_L-Eingang, übernommen.

In Bild 3.2.7. ist ein 4-Bit-Parallel-Rechenwerk gezeigt. Über den Daten-Bus (Informations-Träger) liegen die Daten an den a-Eingängen des Rechenelements an. Die b-Eingänge des Rechenelements sind mit den Ausgängen des Schieberegisters B und den Eingängen der Tri-State-Treiber verbunden. Die

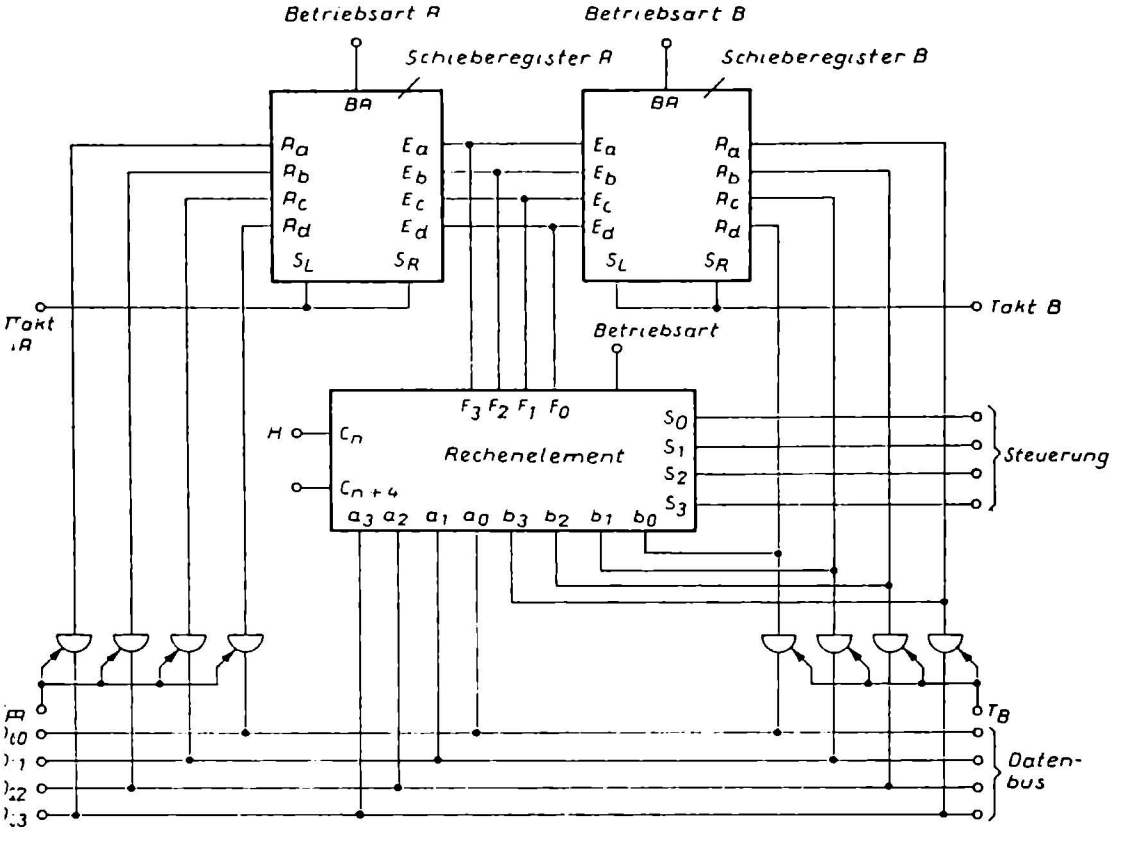


Bild 3.2.7 4-Bit-Parallel-Rechenwerk zum Addieren, Subtrahieren, Multiplizieren und Dividieren

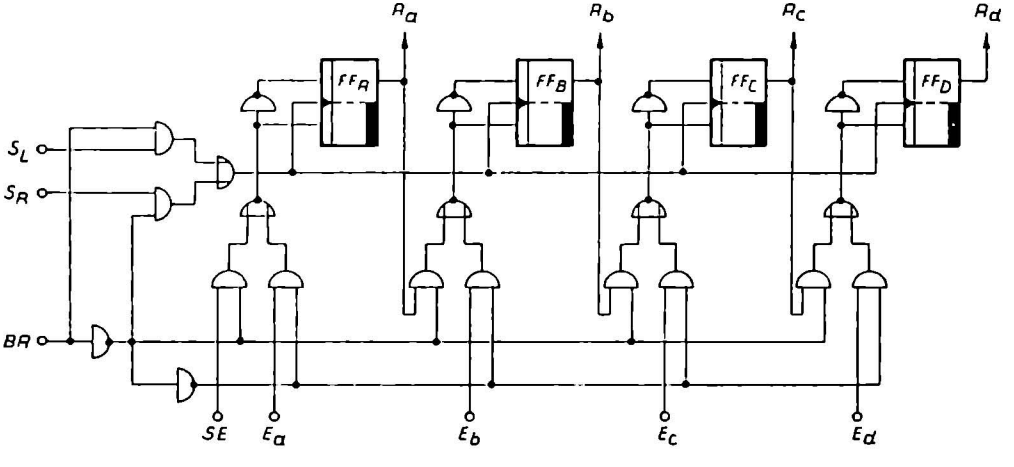


Bild 3.2.8 Schieberegister für Rechts- und Links-Betrieb

F-Ausgänge des Rechenelements sind mit den E-Eingängen der beiden Schieberegister verbunden. Über die Tri-State-Treiber sind die Ausgänge des Schieberegisters A mit dem Daten-Bus verbunden.
Wir können in der Schaltung von Bild 3.2.7 die Informationen über den Daten-Bus aufnehmen und die verarbeiteten Informationen über den Daten-Bus abgeben. Hierzu sind die Tri-State-Treiber erforderlich. Die Steuerung der Treiber geschieht über die T-Eingänge. Bei einem H-Signal an dem T-Eingang leiten die Treiber die Daten durch und schalten sie auf den Daten-Bus. Bei einem L-Signal am T-Eingang ist der Ausgang der Treiber hochohmig, d.h. der Ausgang paßt sich dem Logik-Pegel der Daten-Bus-Leitung an.

Mit den Steuer-Eingängen S_0 bis S_3 werden die einzelnen arithmetischen Funktionen angesteuert. Die beiden Takt-Eingänge S_L und S_R bei den beiden Schieberegistern sind zusammengeschlossen. Mit dem Betriebsart-Eingang können wir die beiden Schieberegister steuern.
Der Übertrags-Eingang bei dem Rechenelement liegt auf H-Signal.

Addition in der Rechenschaltung von Bild 3.2.7:

Folgende Addition soll durchgeführt werden,

10	≙	H L H L
+ 9	≙	H L L H
19	≙	H L L H H

Über den Daten-Bus geben wir den ersten Summanden 10 (HLHL) auf die a-Eingänge des Rechenelementes. An den Steuer-Eingängen muß ein L L L L – Steuerwort aus dem Instruktions-Register anliegen. Dadurch liegt der erste Summand direkt über das Rechenelement an den beiden Schieberegistern an. Die 10 soll nun im Register B (Schieberegister B) zwischengespeichert werden. Wir geben ein H-Signal auf den Betriebsart-Eingang B und auf den Takt B einen Einzelimpuls. Die 10 ist damit gespeichert und liegt über die A-Ausgänge an den b-Eingängen des Rechenelementes an. Durch ein L-Signal an dem T_B-Eingang der Tri-State-Treiber verhindern wir den Rückfluß der gespeicherten Informationen auf den Daten-Bus.
Durch einen Befehl aus dem Instruktions-Register ändert sich das Steuerwort von LLLL auf HLLH (a plus b). Über den Daten-Bus kann jetzt der zweite Summand 9 auf die a-Eingänge des Rechenelementes gegeben werden.
Durch den Befehl HLLH wird die gespeicherte 10 mit der anliegenden 9 addiert.
Wir haben nun zwei Möglichkeiten zur Informations-Weiterverarbeitung. Die erste ist die Speicherung des Ergebnisses durch das Register A und damit bleibt die 10 im Register B erhalten. Die zweite Möglichkeit ist das Ergebnis in das Register B zu speichern und damit können wir eine weitere Addition durchführen.
Bei beiden Möglichkeiten können wir aber durch ein H-Signal an den T-Eingängen der beiden Tri-State-Treiber die Informationen auf den Daten-Bus schalten.

Subtraktion in der Rechenschaltung von Bild 3.2.7:

Folgende Subtraktion soll durchgeführt werden,

11	≙	H L H H
– 9	≙	H L L H
2	≙	L L H L

Diese Subtraktion können wir auf drei Arten durchführen,
a) mit einem Subtrahierwerk
b) mit dem B-Komplement in einem Addierwerk und
c) mit dem B-1-Komplement in einem Addierwerk.
Alle drei Möglichkeiten können einfach realisiert werden,

a)

11	≙	H L H H
− 9	≙	H L L H
2	≙	L L H L

Durch das Subtrahierwerk wird die Differenz zwischen 11 und 9 hergestellt. Da das Rechenelement von Bild 3.2.7 kein Subtrahierwerk aufweist, können wir eine Subtraktion auf dieser Art nicht durchführen.

b) Diese Möglichkeit der indirekten Subtraktion wird gerne angewandt.

11	≙	H L H H	bleibt	H L H H
9	≙	H L L H	Kompl.	+ L H H L
				H L L L H
				+ ⟵ H
				L L H L

Die 9 wird komplementiert, d.h. H L L H wird zu L H H L. Durch die Komplementbildung können wir nun die beiden Zahlen addieren. Der Übertrag muß um vier Stellen nach links geschoben werden und es erfolgt eine weitere Addition.
Durch die zweite Addition, der sogenannten Korrektur-Addition ist das Verfahren nach dem B-Komplement viel zu umständlich, da der Übertrags-Ausgang auf den Übertragseingang zurückgeschaltet werden muß.

c) Das B-1-Komplement läßt sich mit dem Rechenelement von Bild 3.2.7 einfach durchführen.

11	≙	H L H H							H L H H
9	≙	H L L H	⇒	9 − 1	⇒	H L L L	⇒ Kompl.	⇒	+ L H H H
				B−1		8			L L H L

Durch das B-1-Komplement benötigen wir keine Korrektur-Addition. Das Rechenelement von Bild 3.2.7 ist für das B-1-Komplement ausgelegt.

Mit einem Steuerwort LLLL wird die 11, die über den Daten-Bus anliegt, auf die beiden Register geschaltet. Das Register A übernimmt die Information und speichert sie ab. Das Steuerwort ändert sich durch einen Instruktions-Befehl von LLLL auf LLHH. Über den Daten-Bus liegt die 9 am Rechenelement an. Durch das Steuerwort LLHH wird nun aus der 9 das B-1-Komplement gebildet, d.h. das Rechenelement subtrahiert von 9 eine 1 und bildet dann das Komplement. Im Dual-Code sieht dies folgendermaßen aus:

9	≙	H L L H	
9 − 1	≙	H L L L	⇐ B − 1
		↑ ↑ ↑ ↑	
Kompl.	≙	L H H H	

Das Rechenelement bildet durch das Steuerwort LLHH direkt das B-1-Komplement aus den anliegenden Eingangsdaten. Damit liegt an den Ausgängen des Rechenelementes das B-1-Komplement an den beiden Registern an. Durch einen Speicher-Befehl speichert das Register B das B-1-Komplement.
Wir ändern durch ein neues Steuerwort die arithmetische Operation des Rechenelementes. Das Steuerwort lautet: HLLH (addiere). Über den Tri-State-Treiber T_A geben wir die gespeicherte 11 auf den Datenbus und damit auf die a-Eingänge des Rechenelementes. Das Rechenelement führt eine Addition zwischen den a- und b-Eingängen aus und an den Ausgängen liegt das Ergebnis der Addition vor. Wir können nun im Register A das Ergebnis, also die Differenz zwischen 11 und 9 speichern.

Multiplikation in der Rechenschaltung von Bild 3.2.7:

Folgende Multiplikation ist durchzuführen,

```
   2   ·    7        =    14
LLHL · LHHH
-----------
        LLHL
       LLHL
      LLHL
     LLLL
    -------
    LLLHHHL
```

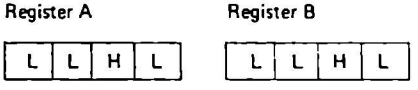
Da wir eine 4-Bit-Recheneinheit haben, so ergibt die Multiplikation nur ein vierstelliges Ergebnis und zwar H H H L $\hat{=}$ 14.
Über den Daten-Bus geben wir die 2 auf das Rechenelement. Durch den Steuerbefehl LLLL geben wir die 4 direkt auf die Register und speichern die 4 in Register A und B.

Register A

L	L	H	L

Register B

L	L	H	L



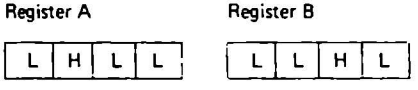
Durch einen Steuerbefehl ändern wir die arithmetische Operation des Rechenwerkes. Wir geben das Befehlswort HHLL, d.h. die a-Eingänge des Rechenwerkes werden zur nächst höherwertigen Stelle nach links verschoben. Über die Tri-State-Treiber geben wir die gespeicherte 4 des Registers A auf das Rechenwerk.

Register A

L	H	L	L

Register B

L	L	H	L



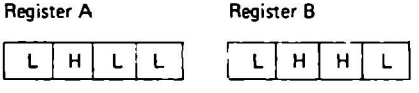
Im Register A steht nun die Information, die um ein Bit nach links verschoben ist. Das Steuerwort ändert sich auf HLLH und der Tri-State-Treiber T_A gibt Register A frei. Durch die Addition ergibt sich ein Zwischenergebnis, daß wir in Register B speichern.

Register A

L	H	L	L

Register B

L	H	H	L



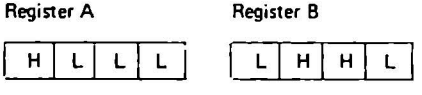
Wir ändern erneut das Befehlswort auf HHLL und geben die Informationen von Register A frei. Im Rechenelement erfolgt eine Verschiebung nach links und dadurch ergibt sich folgender Stand in beiden Richtungen.

Register A

H	L	L	L

Register B

L	H	H	L



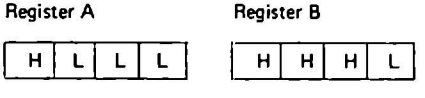
Das Befehlswort ändert sich auf HLLH und der Tri-State-Treiber T_A gibt Register A frei. Im Rechenelement wird Register A mit Register B addiert und das Ergebnis speichern wir in Register B.

Register A

H	L	L	L

Register B

H	H	H	L



Im Register B steht das Ergebnis der Multiplikation.

Der wesentliche Vorteil von der Schaltung in Bild 3.2.7 ist die Austauschbarkeit der beiden Register untereinander mit Zwischenoperationen durch das Rechenelement.

Division in der Rechenschaltung von Bild 3.2.7:

Folgende Division soll durchgeführt werden,

```
   15          5     =     3
  H H H H    L H L H  =  L L H H
- H L H
-------
  L H L H
- L H L H
-------
  L L L L
```

Mit dem Rechenelement von Bild 3.2.7 kann diese Art der Division nicht durchgeführt werden, da das Rechenelement nicht jedes Bit zur nächst niedrigen Stelle schieben (also ein Rechtsschieben der Information) und außerdem nicht subtrahieren kann.
Wir verwenden daher ein anderes Verfahren. Über den Daten-Bus geben wir die 15 in das Register A, wenn das Steuerwort LLLL anliegt. Über den Instruktions-Befehl ändert sich das Steuerwort auf LLHH. Über den Daten-Bus legen wir nun die 5 an. Dadurch ergibt sich am Ausgang des Rechenelementes das B-1-Komplement und dies speichern wir im Register B.

Register A				Register B			
H	H	H	H	H	L	H	H

Das Steuerwort schaltet auf HLLH (addiere) um und gibt die gespeicherten Informationen des Registers A auf den Daten-Bus. Die Eingangsinformationen werden von dem Rechenelement verarbeitet und das Ergebnis im Register A gespeichert.

Register A				Register B			
H	L	H	L	H	L	H	H

Im Register A steht das Zwischenergebnis der ersten Subtraktion, das wir mittels dem B-1-Komplement ausgeführt haben.
Eine externe Vergleicher-Schaltung untersucht nun, ob der invertierte Inhalt des Registers A kleiner ist als der Inhalt von Register B. Da aber dies nicht der Fall ist, wird eine erneute Subtraktion eingeleitet.

```
Register A:     H L H L
Register B :  + H L H H
              -------
                L H L H
```

Die verarbeitete 5 (LHLH) speichern wir in Register A und führen mit einer externen Logik einen Vergleich mit den beiden Inhalten der Register durch. Da der Inhalt des Registers A nicht kleiner als der invertierte Inhalt des Registers B ist, kann das Rechenelement eine erneute Subtraktion ausführen.

```
Register A:     L H L H
Register B:   + H L H H
              -------
                L L L L
```

Die verarbeitete 0 (LLLL) speichern wir in Register A und führen mit der externen Logik einen erneuten Vergleich durch. Da der Inhalt von Register A kleiner ist als der von Register B, so ist der Rechenvorgang für die Division beendet.
Haben wir eine Division, bei der ein Rest verbleibt, so ist dieser in Register A gespeichert.

Weitere Schaltungsmöglichkeiten von Bild 3.2.7:

Mit der Schaltung können noch weitere Funktionen durchgeführt werden. Ein typisches Beispiel ist der Transport von Daten aus dem Register A über die Tri-State-Treiber T_A und dem Rechenelement in das Register B. Andererseits können die Daten von Register B über die Tri-State-Treiber T_B und dem Rechenelement in das Register A transportiert werden.

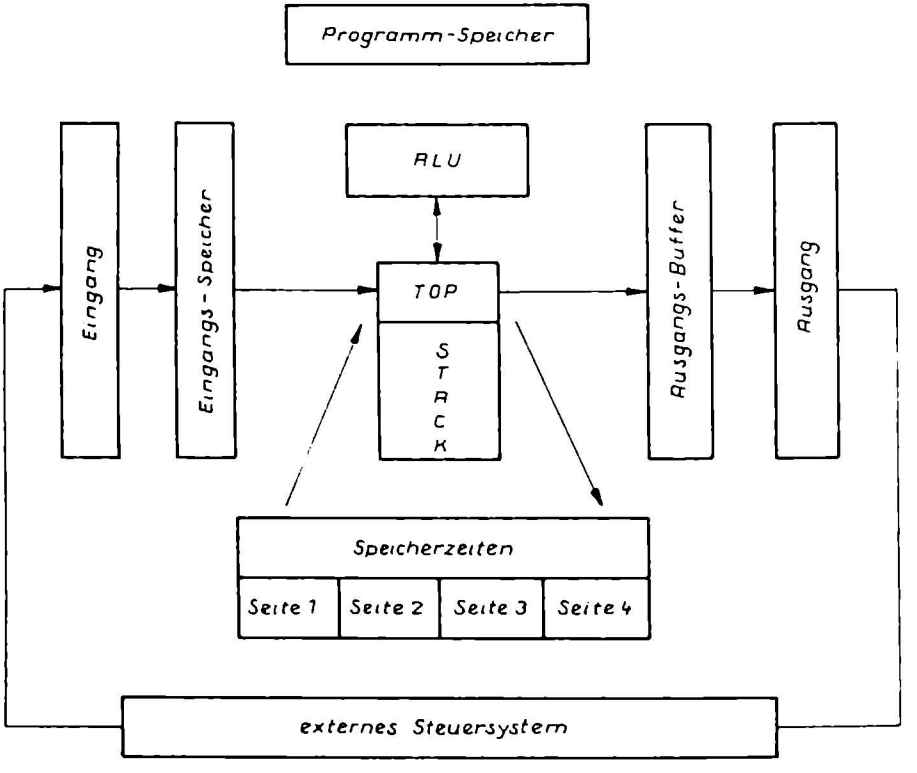
Bei dem Durchlauf der Daten durch das Rechenelement können wir auch logische Funktionen durchführen, z.B. wir invertieren die Eingänge a des Rechenelementes zu a und speichern die invertierten Informationen des Daten-Busses in einem der beiden Register.

Eine weitere Möglichkeit ist der Vergleich der beiden Speicherinhalte über die logische Betriebsart. Bei einem Steuerbefehl LHHL wird zwischen den beiden Registerinhalten ein Äquivalenz-Vergleich nach der Booleschen Algebra durchgeführt. Bei dem Steuerbefehl HLLH erfolgt der Antivalenz-Vergleich nach der Booleschen Algebra. Das Vergleichs-Verfahren hat aber einen Nachteil. Einer der beiden Register-Inhalte muß dabei gelöscht werden, da sonst das Ergebnis des Vergleiches nicht auf den Daten-Bus gegeben werden kann.

Soll der Registerinhalt mit dem Daten-Bus verglichen werden, so speichern wir erst eine Information in das Register B. Danach geben wir eine neue Information auf den Daten-Bus und speichern das Ergebnis des Vergleiches in dem Register A ab. Über die Tri-State-Treiber können wir das gespeicherte Ergebnis des Vergleiches anschließend auf den Daten-Bus geben.

4. Mikroprozessoren Typenzusammenstellung

4.1 SBA Mikroprozessor
(Sequential Boolean Analyzer)



Innenschaltung des SBA - Mikroprozessors

SBA-Mikroprozessor

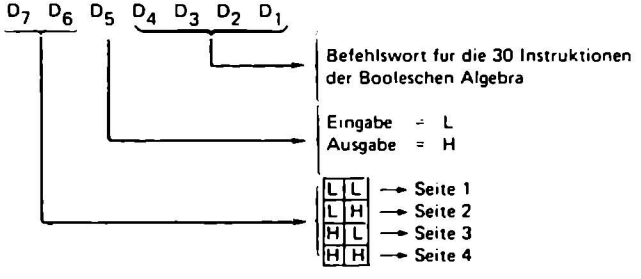
Die Firma General Instruments stellt einen einfachen 1-Bit Mikroprozessor für die Anwendungen der Booleschen Algebra her. Durch den einfachen Aufbau kann der Baustein nur für bestimmte Anwendungen verwendet werden.

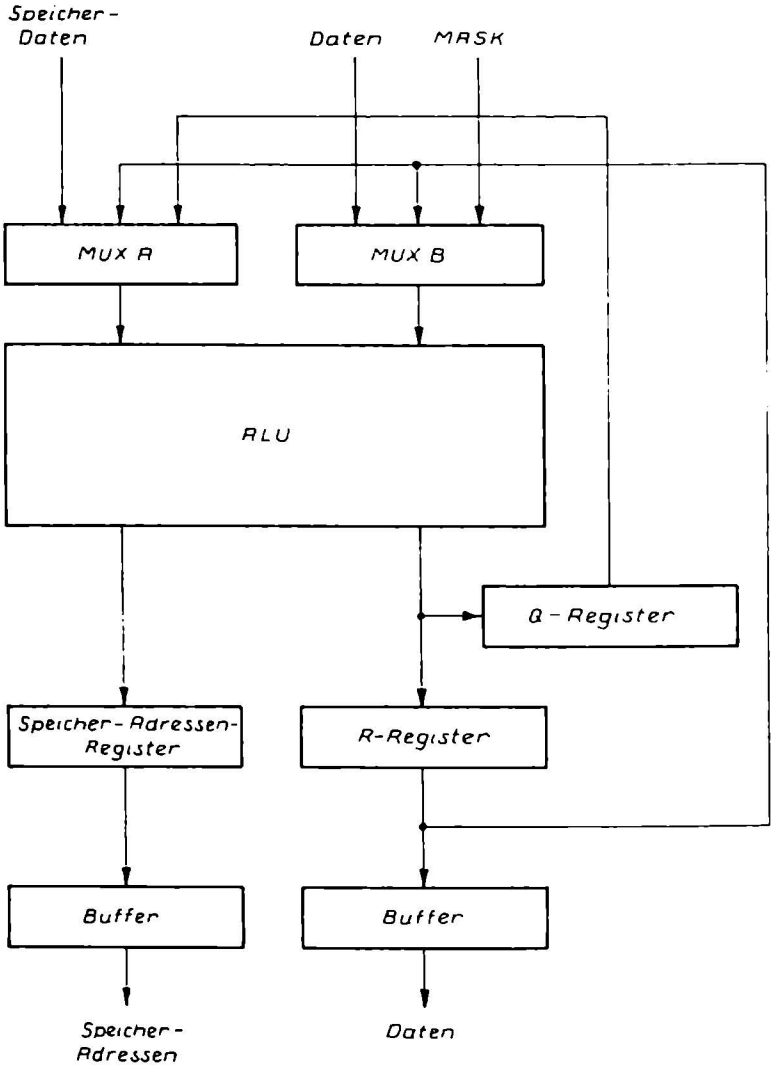
Der Baustein hat einen Programm-Speicher für die einzelnen Booleschen Funktionen. Es ergeben sich 30 verschiedene Möglichkeiten der Booleschen Algebra.

Die Daten für den SBA-Baustein werden im Eingangsspeicher gespeichert und liegen an dem TOP an. In dem TOP können diese Eingangsinformationen mit den Speicherseiten verknüpft werden. Hierzu gibt TOP die Informationen an die ALU weiter, die aber keine arithmetischen Befehle direkt ausführen kann. Die verknüpften Informationen gelangen entweder über das TOP wieder in eine Speicherseite zurück oder sie werden über den Ausgangs-Buffer auf den Ausgang geschaltet.

Wesentlich bei dem SBA-Baustein sind die internen Speicherseiten. Dadurch können Informationen in den einzelnen Seiten zwischengespeichert werden. Durch einen Befehl auf den Programm-Speicher wird die einzelne Seite freigegeben und im TOP eingelesen. Über dem TOP entsteht die Verknüpfung in der ALU. Durch den Befehl auf den Programm-Speicher kann die Betriebsart beim Auslesen festgelegt werden.

Aufbau eines Befehls:





Innenschaltung des 2-Bit-STTL-Mikroprozessors

2-Bit-STTL-Mikroprozessor

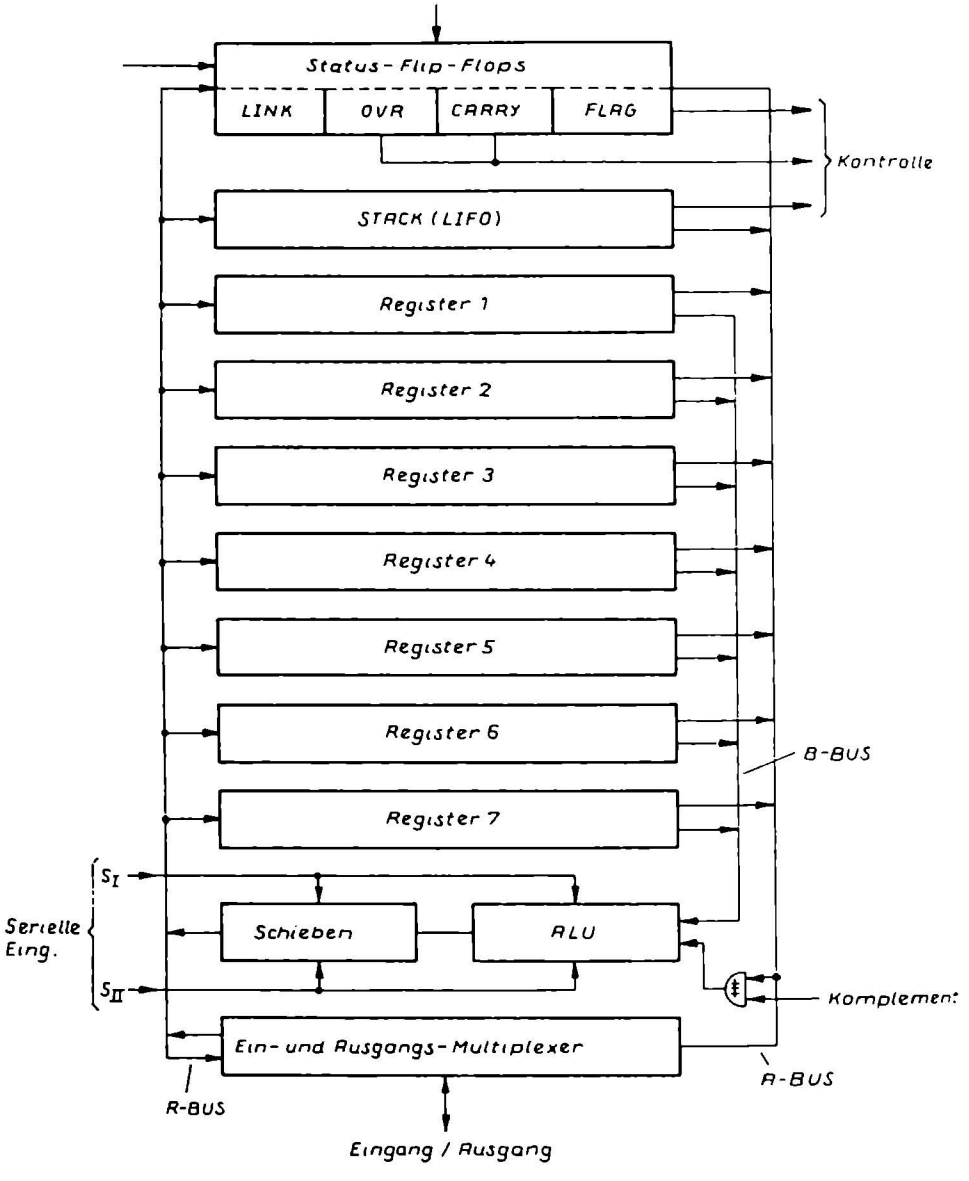
Die Firma Intel liefert einen 2-Bit-Mikroprozessor fur besondere Anwendungsfalle. Es gibt hierzu folgenden Baustein:

D 3001: Mikroprogramm-Kontroller
D 3002: 2-Bit-Mikroprozessor
D 3003: Übertrags-Einheit
D 3212: Buffer-Baustein
D 3214. Unterbrechungs-Kontroller
D 3216: BUS-Treiber-Baustein
D 3226: BUS-Treiber-Baustein

Durch die 2-Bit-Struktur ergibt sich ein begrenzter Einsatz in der industriellen Elektronik. Durch die Ubertrags-Einheit laßt sich der Baustein aber einfach auf die gewunschte Datenlänge zusammenbauen. Dadurch kann die Bitlänge für jeden Anwendungsfall individuell zusammengestellt werden. Die Adressierung der externen Baugruppen erfolgt bei diesem System nicht über den Mikroprozessor, sondern uber den Mikroprogramm-Kontroller. Es ergibt sich ein direkter Adressierbereich von 512 Worten.

Die Befehlslänge des Mikroprozessor-Bausteines D 3002 umfaßt 7-Bits. Hierzu ergeben sich über 40 Befehle. Die Zugriffszeit für die ALU-Befehle dauert 150ns. Dadurch ergibt sich eine maximale Taktfrequenz von 6,67MHz. Da der Baustein in Schottky-Transistor-Transistor-Technologie aufgebaut ist, kann der Mikroprozessor auch im reinen Gleichstrombetrieb gefahren werden.

An den beiden Daten-Eingangen liegen die Informationen. Über die beiden vorgeschalteten Multiplexer erhalt die ALU ihre Daten. Hierzu ergeben sich für die Ausgänge folgende wichtige Funktionen. Uber das Speicher-Adressen-Register und dem nachgeschalteten Buffer können bestimmte Speicher-Adressen in dem Mikroprogramm-Kontroller adressiert werden. Die andere Möglichkeit ist die Datenausgabe über die beiden Register. Das Q-Register verbindet die ALU mit dem Multiplexer MUX A. Das R-Register ist mit dem Ausgang und mit den beiden Multiplexern MUX A und MUX B verbunden. Dadurch ergeben sich zahlreiche Möglichkeiten der Verknüpfungen. Während das Q-Register mit den Daten des Multiplexers MUX B logisch oder arithmetisch verknüpft wird, bietet das R-Register in Verbindung mit den beiden Multiplexern weit mehr Möglichkeiten. Neben einer Verknupfung mit dem Q-Register sind auch Verbindungen mit dem Speicher-Daten-Eingang, dem Daten-Eingang und dem MASK-Eingang möglich.
mit FORTRAN IV identisch, wenn das System auf 16-Bit erweitert worden ist
Für das System 3000 wurde eine eigene Programmiersprache, die CROMIS entwickelt. Die CROMIS ist weitgehend mit FORTRAN IV identisch, wenn das System auf 16-Bit erweitert worden ist.



Innenschaltung des 4-Bit-Mikroprozessor-Slices

Die Firma National Semiconductor stellt einen 4-Bit-Mikroprozessor her, der einfach erweiterbar ist, den IMP-4.

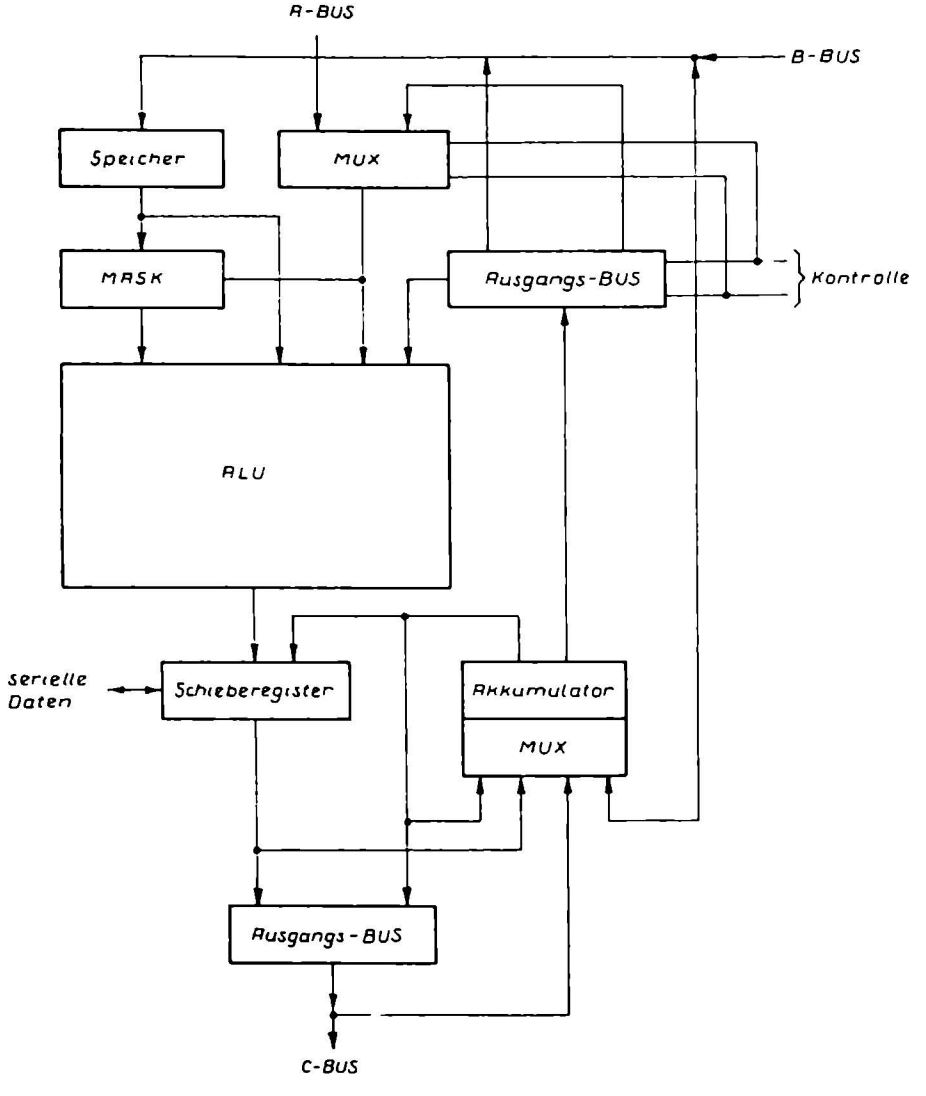
Der Mikroprozessor besteht aus den Status-Flip-Flops, einem Stack, sieben Register, einem Schieberegister, der ALU und dem Ein- und Ausgangs-Multiplexer. Die einzelnen Baugruppen sind mit drei internen BUS-Systemen verbunden. Der R-BUS übernimmt aus dem Ein- und Ausgangs-Multiplexer die Daten und Befehle und gibt sie an die einzelnen Baugruppen weiter. Dabei besteht nur ein Einrichtungsbetrieb, außer bei der Ein/Ausgabe. Der Mikroprozessor kann Daten aufnehmen und sie wieder ausgeben, ohne daß eine Zwischenspeicherung in einer der internen Baugruppen vorgenommen wurde. Die beiden anderen BUS -Systeme verbinden die einzelnen Baugruppen mit der ALU und mit dem Ein- und Ausgangs-Multiplexer.

Mit einem Eingabe-Befehl übernimmt der Ein- und Ausgangs-Multiplexer die Eingangsinformationen und transportier sie an eines der Register weiter. Fur eine Adresse eines Unterprogramm-Sprunges ist der STACK oder Kellerspeicher zuständig. Der STACK ist als LIFO ausgebildet, d.h. last in - first out Dadurch kann eine Prioritats-Unterprogrammierung vorgenommen werden. Die letzte eingeschriebene Adresse wird immer zuerst ausgelesen. Danach die zweitletzte usw.
Die Register in dem Mikroprozessor konnen die Eingangsinformationen zwischenspeichern. Durch einen Befehl werden sie entweder auf den A- oder B-Bus geschaltet. Der Mikroprozessor kann gleichzeitig uber die beiden BUS-Systeme Informationen befördern. Diese Informationen stehen an der ALU an. In der ALU werden die arithmetischen oder logischen Funktionen durchgefuhrt. Das Auslesen der Informationen erfolgt über ein Schieberegister wieder zuruck in eines der Register.
Sollen Daten ausgelesen werden, so ist dies nur uber den A-BUS moglich. Informationen der ALU können nur entweder uber eines der Register, oder uber den R BUS in den Ein- und Ausgangs-Multiplexer ausgelesen werden.

Mit dem Eingang "Komplement" wird der A-BUS entweder direkt oder negiert auf den Eingang der ALU gegeben. Es gilt folgende Funktionstabelle

0	→	direkt
1	→	negiert

Der Eingang "Komplement" steuert ein Exklusiv-ODER- Gatter an. Dadurch entsteht ein Komplement.



Innenschaltung des 4-Bit-ECL-Mikroprozessors

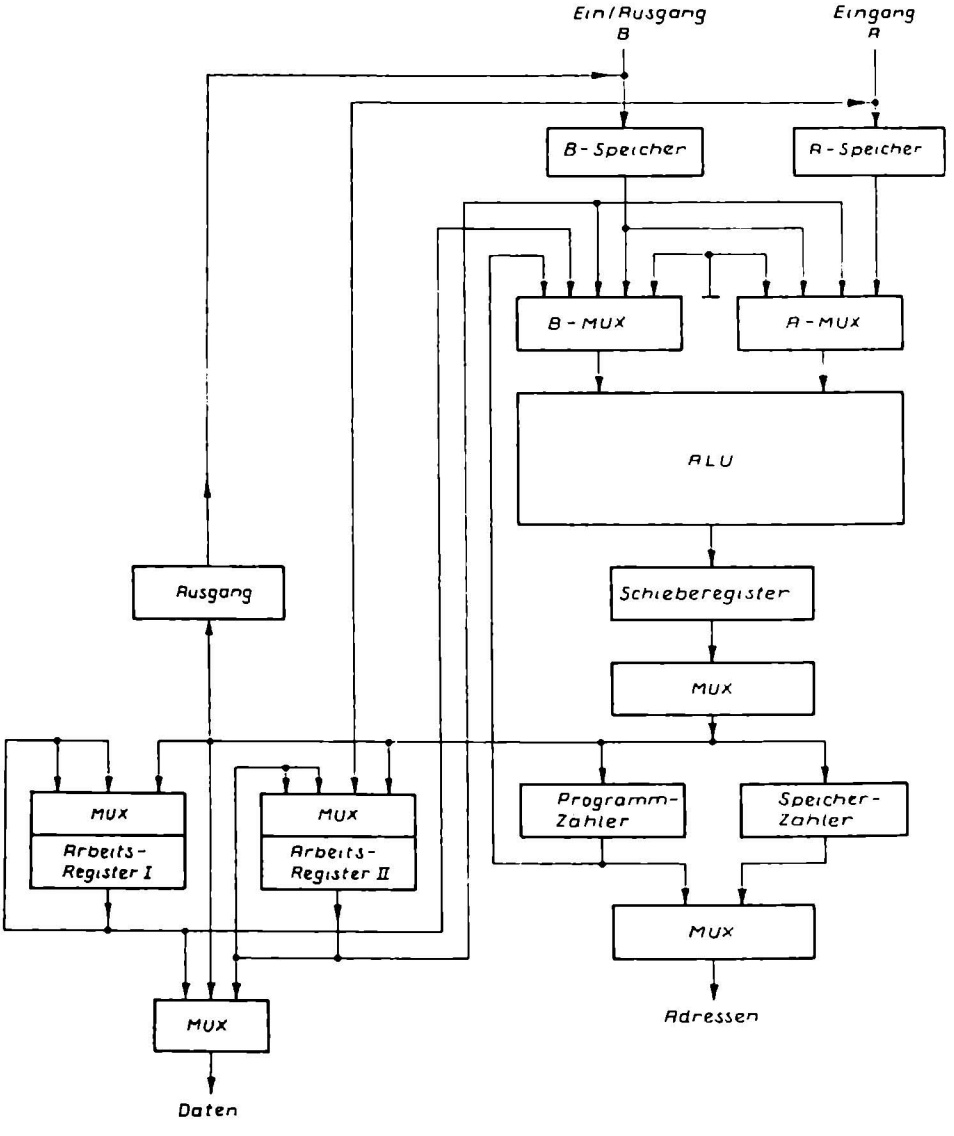
4-Bit-ECL-Mikroprozessor

Die Firma Motorola Semiconductor stellt einen sehr schnellen 4-Bit-Mikroprozessor in ECL-Technologie her. Es handelt sich um den Typ 10800. Für diesen Baustein gibt es folgende Hardware:

MC 10800: 4-Bit-Mikroprozessor
MC 10801: Mikroprogramm-Kontroller
MC 10802: Taktgenerator
MC 10803: Speicher-Interface-Schaltung
MC 10804: 4-Bit-Schieberegister
MC 10805: 5-Bit-Schieberegister
MC 10806: 32 x 9-Bit-Speicherregister
MC 10808: 16-Bit-programmierbares Schieberegister

Durch die Struktur des MC 10800 kann eine einfache Erweiterung vorgenommen werden. Bei einer Parallelschaltung von zwei Bausteinen ergibt sich ein 8-Bit-Mikroprozessor. Hierzu ist der Übertrags-Ausgang mit dem Übertrags-Eingang der einzelnen Stufen zu verbinden.
Der Mikroprozessor MC 10800 kann nur in Verbindung mit dem Mikroprogramm-Kontroller arbeiten. Dabei kontrolliert der Baustein MC 10801 den Mikroprozessor. Ist der Befehlsablauf in dem Baustein MC 10800 abgeschlossen, so meldet er dies mit MC 10801. Danach holt der Mikroprogramm-Kontroller den nächsten Befehl für den Mikroprozessor ein.

Die Eingabe der Daten erfolgt über den A-, B- und C-BUS. Der A-BUS liegt direkt an dem Multiplexer MUX an. Mit einem Datenwort an dem Kontroll-Eingang wird bestimmt, ob der A-BUS oder Ausgangs-BUS anliegt. Der Ausgang des MUX ist direkt mit der ALU verbunden. Damit können über den A-BUS direkte Daten eingeschrieben werden.
Die andere Möglichkeit der Dateneingabe ist über den B-BUS. Der B-BUS ist mit dem Eingangsspeicher verbunden. Damit können über den A-BUS direkte Daten eingeschrieben werden.
Die andere Möglichkeit der Dateneingabe ist über den B-BUS. Der B-BUS ist mit dem Eingangsspeicher verbunden. Durch ein Signal werden die B-BUS-Informationen in dem Speicher gespeichert. Der Ausgang des Speichers liegt direkt an der MASK und der ALU. Der MASK ist ein Decoder, der Befehle des B-BUS decodieren kann und damit die ALU steuert. Arbeitet der MASK, werden die Daten des MUX mit den Daten des Ausgangs-BUS verknüpft oder einer arithmetischen Funktion unterworfen.
Arbeitet der MASK nicht, liegen die Daten des B-BUS über den Speicher an der ALU und werden dort mit den Daten des A-BUS arithmetisch oder logisch verknüpft.
Die ALU hat 10 Funktionen. Diese Funktionen beschränken sich nur auf die direkten ALU-Funktionen. In Verbindung mit den anderen Einheiten steigt aber der Befehlsvorrat auf ca. 80 Befehle an.
Die Ausgänge der ALU liegen an dem Schieberegister. Damit können die Daten seriell ausgegeben werde. Bei einer parallelen Ausgabe kann dies über den Ausgangs-BUS erfolgen.

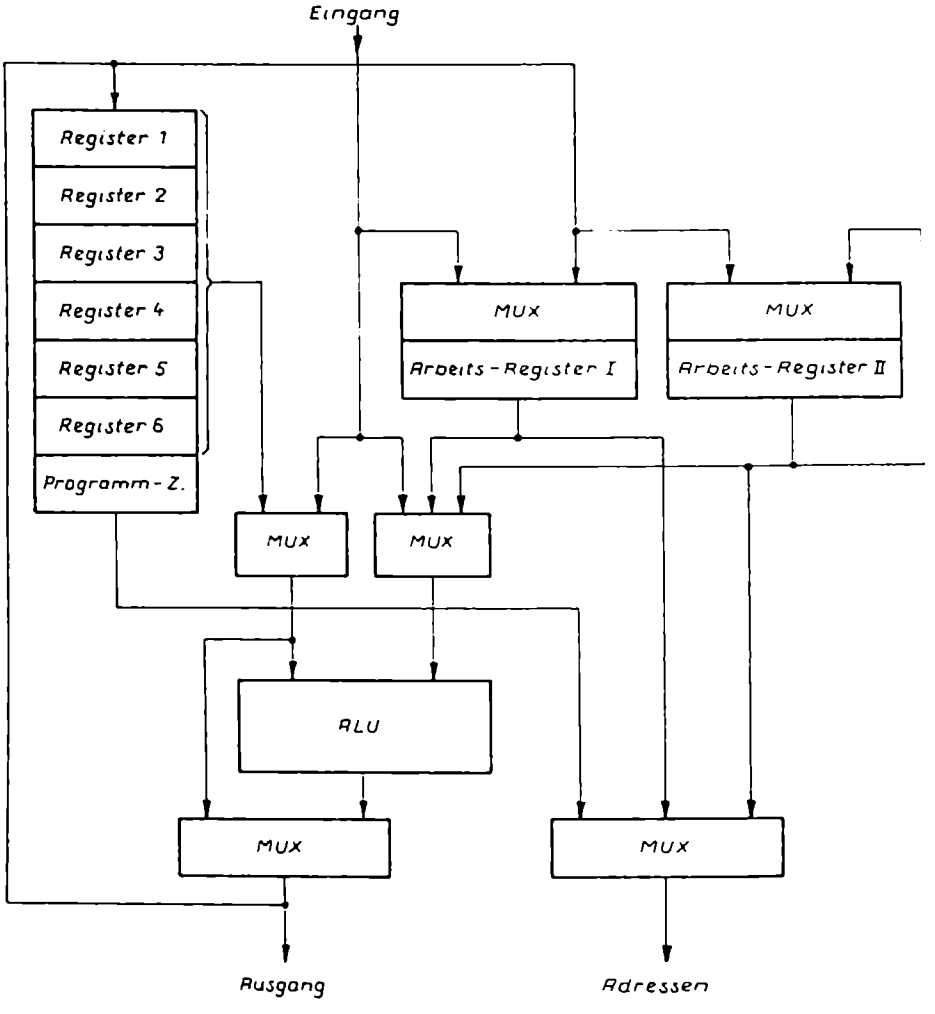


Innenschaltung des 4 Bit-STTL-Mikroprozessor-Slices

Die Firma Texas Instruments stellt einen 4-Bit-Mikroprozessor in Schottky-TTL-Technologie her. Es handelt sich um den Baustein SN74S481. Zu diesem erweiterbaren Baustein gibt es noch folgende Bausteine:

SN74S481:	4-Bit-Mikroprozessor-Slice
SN74S482:	4-Bit-Kontroller
SN74S330:	Freiprogrammierbare Logikanordnung PLA
SN74S225:	16-Wort-mit 5-Bit-FIFO
SN74S226:	Vierfach BUS-Sender
SN74S182:	Übertragseinheit
SN74S240:	Buffer-Baustein
SN74S241:	Buffer/Treiber-Baustein
SN74S373:	Speicher-Baustein
SN74S374	D-Flip-Flop-Register

Der Baustein hat zwei Eingänge A und B, wobei der Eingang B auch als Ausgang verwendet werden kann. Damit muß die Baugruppe Ausgang einen Tri-State-Ausgang aufweisen.
Über die beiden Eingänge A und B werden die Eingangsinformationen in die beiden Speicher eingegeben. Der A- und der B-Multiplexer verbinden jeweils zwei Funktionseingänge mit der ALU. Die ALU selbst kann 16 Befehle ausführen. Diese Befehle zergliedern sich in arithmetische und logische.
Der Ausgang der ALU ist mit einem Schieberegister verbunden. Durch Schiebebefehle kann pro Taktimpuls die Wertigkeit um eine Stelle nach links oder rechts geschoben werden. Die Wertigkeiten können auch nur zwischengespeichert anliegen und werden über den Multiplexer MUX auf den internen Daten-BUS geschaltet. Dieser transportiert sie entweder zum Speicher-Zähler, Programm-Zähler, in das Arbeits-Register I oder II oder zum Ausgang.
Der Speicherzähler gibt über den Multiplexer direkt die Adressen auf den externen Adressen-BUS aus. Der Programmzähler gibt seine Informationen entweder ebenfalls auf den MUX und dann weiter auf den Adressen-BUS oder über den B-MUX in die ALU ein.
Die beiden Arbeits-Register I und II dienen zum Zwischenspeichern von Informationen. Die Ausgänge der beiden Arbeitsregister sind mit dem Daten-Multiplexer verbunden. Das Arbeits-Register I kann über den B-MUX seine gespeicherten Daten wieder in die ALU eingeben. Dagegen ist man mit dem Arbeits-Register II in der Lage, über den A- oder den B-MUX in die ALU die gespeicherten Daten einzuschreiben. Bei einer Adressierung der beiden Arbeits-Register kann zwischengespeichert und später arithmetisch oder logisch verknüpft werden.
Die Taktfrequenz des Bausteines liegt zwischen DC und 15MHz. Damit kann der Mikroprozessor auch statisch arbeiten.



Innenschaltung des 4-Bit I²L-Mikroprozessors

4-Bit-I^2L-Mikroprozessor

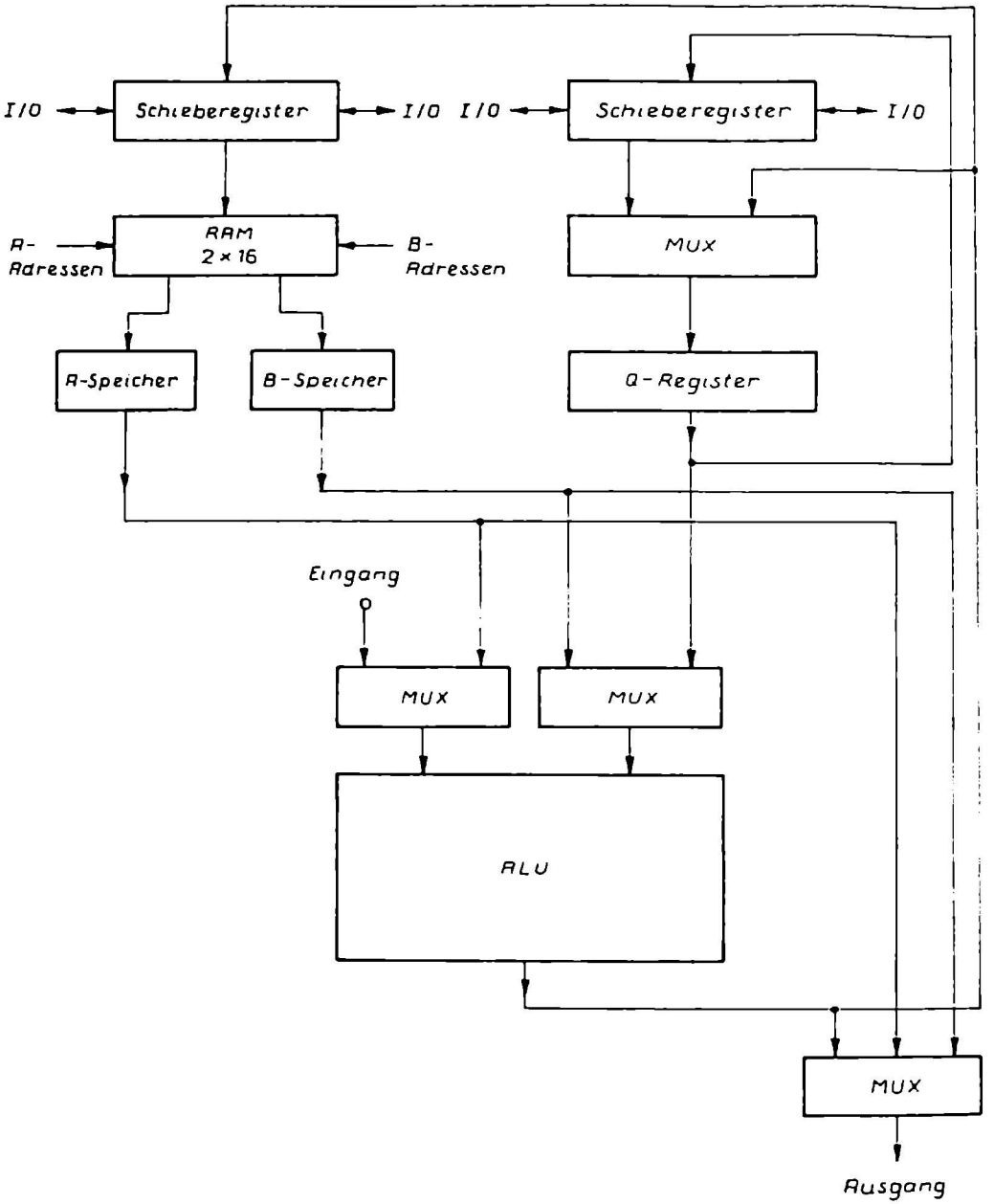
Die Firma Texas Instruments stellt den Mikroprozessor SBP0400A, bzw. den SBP0401A her. Durch die I^2L-Technologie haben diese Bausteine einen sehr schnellen Zugriff und eine geringe Leistungsaufnahme. Der Unterschied zwischen den beiden Typen liegt an einem Pipeline-Register. Der Baustein SBP0400A hat ein internes Pipeline-Register auf dem Chip. Der Baustein SBP0401A benotigt ein externes Pipeline-Register.

Die ALU erhalt die Eingangs-Daten uber den Eingang, die intern gespeicherten Daten uber die beiden Arbeits-Register I und II oder direkt aus den sechs Registern. Dadurch ergibt sich ein universeller Rechenbetrieb. Die Ausgabe der Daten erfolgt uber den Ausgang, die Datenspeicherung in einem der sechs Register oder in den beiden Arbeits-Registern I und II. Die Ubernahme oder die Abspeicherung von Informationen ist von dem externen Taktsignal abhangig. Die Taktfrequenz reicht vom reinen Gleichstrombetrieb bis zu einer maximalen Taktfrequenz von 3,3MHz.

Die Architektur des Mikroprozessors SBP0400A und SBP0401A ist sehr einfach. Die 16-Funktionen-ALU wird von einem 4-Bit-Wort angesteuert. Der Befehlsvorrat reicht dabei von den einfachen logischen Verknupfungen bis zu arithmetischen Funktionen. Das 4-Bit-Wort zur Befehlsansteuerung kann durch eine externe TTL-Logik erzeugt werden. Durch die I^2L-Technologie sind keine besonderen Interface-Schaltungen erforderlich, da die Betriebsspannung U_b = 5V betragt.

Die externe Adressierung durch den Mikroprozessor findet mit dem Programm-Zahler PC (programm counter) statt. Die Adressenausgabe erfolgt uber den Adressen-Multiplexer MUX. Fur den normalen Programmablauf liegen die Adressen des Programm-Zahlers direkt an dem Adressen-Ausgang. Tritt ein Sprungbefehl auf, so erfolgt die Adressenausgabe uber eines der beiden Arbeits-Register I und II. Damit ist dieser Mikroprozessor-Baustein in der Software sehr einfach.

Die Ubertragung einer Sprungadresse in eines der beiden Arbeits-Register kann entweder direkt uber den Eingang oder durch eine Zwischenspeicherung in eines der sechs Register erfolgen. Mit einem Befehl wird der Stand des Programmzahlers in eines der Register abgespeichert. Anschließend nimmt der Programm-Zahler die Sprung-Adresse auf und kann auf dieser weiter arbeiten. Mit einem Rucksprung aus dem Unterprogramm wird die in dem Register abgespeicherte Adresse wieder in den Programm-Zahler eingeschrieben. Danach nimmt der Mikroprozessor sein ursprungliches Programm wieder auf.



Innenschaltung des Mikroprozessor-Slices 5701

Die Firma Monolithic Memories stellt die beiden Mikroprozessor-Slice 5701 und 6701 her. Der Unterschied zwischen diesen beiden Typen ist die Spezifikation. Der Typ 5701 ist eine militärische Version und der 6701 die kommerzielle Version.

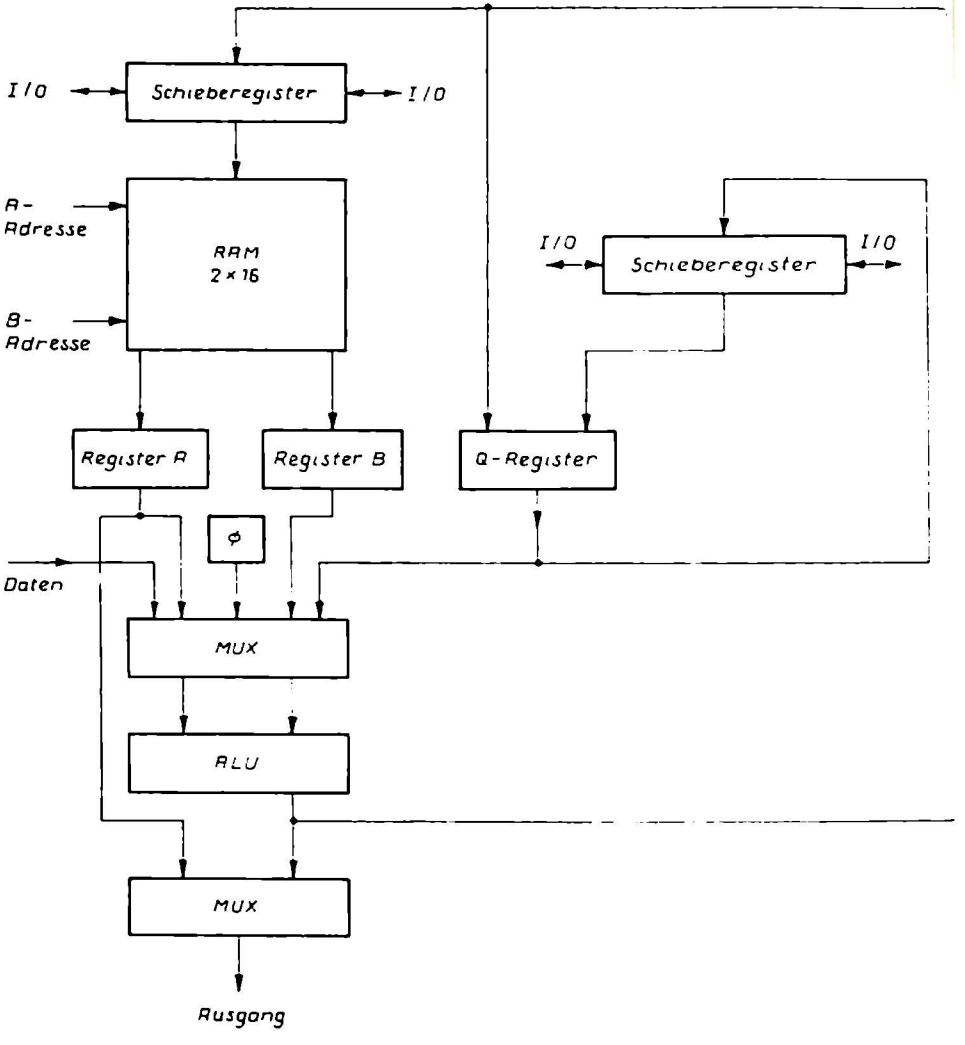
Die Architektur dieser Mikroprozessoren besteht aus einer 36-Funktionen-ALU, einem zweifach-RAM mit je 16 Speicherplätzen, einem Q-Register und zwei Schieberegistern. Damit ergibt sich ein universeller Rechenbetrieb.

Die 36-Funktionen der ALU werden aus einem 8-Bit-Wort gebildet. Dieses Bit-Wort steuert ein internes ROM mit 352 Speicherplätzen an. Dadurch entstehen aus dem 8-Bit-Wort die 36 Funktionen. Neben den Additions-, Subtraktions-, Increment-, Decrement- und logischen Befehlen können auch Transport-Befehle durchgeführt werden.

Die Software kann nach dem FORTRAN IV-CROSS-ASSEMBLER erfolgen. Durch eine Erweiterung des Mikroprozessors ergibt sich eine 16- oder 32-Bit-Rechenanlage, die dann direkt über den FORTRAN IV arbeiten kann. Die Erweiterung erfordert keine Übertrags-Einheiten, wenn kein schneller Rechenbetrieb durchgeführt wird. Bei der Verwendung von Mikroprogramm-Sequenzern vom Typ 57110 (militärische Version) oder dem Typ 67110 (kommerzielle Version ergibt sich ein gemischter paralleler/serieller Betrieb, wodurch die Rechengeschwindigkeit erhöht wird.

Durch die STTL Technologie (Schottky-Transistor-Transistor-Logik) ergibt sich ein sehr schneller Betrieb. Die maximale Rechengeschwindigkeit liegt bei einer 4-Bit-Version bei 5,2 MHz. Da es sich um eine verbesserte TTL-Technik handelt, ergeben sich auch alle Vorteile der TTL-Bausteine, wie der statische Betrieb. In Verbindung mit dem Mikroprogramm-Sequenzer läßt sich ein statischer Mikroprozessor einfach realisieren. Der Mikroprogramm-Sequenzer kontrolliert den Arbeitsvorgang des Mikroprozessors. Damit tritt zwar eine Begrenzung in der schnellen Arbeitsweise auf, aber der Programmablauf wird kontinuierlich gesteuert.

Die Eingabe der Daten erfolgt über den Eingang parallel. Diese Daten liegen über den Multiplexer direkt an der ALU. Von dem Ausgang der ALU gelangen die Daten über einen weiteren Multiplexer auf den Ausgang oder werden über ein Schieberegister in das RAM eingeschrieben. Durch dieses Schieberegister können Daten seriell eingeschrieben oder seriell ausgelesen werden. In dem RAM erfolgt die Zwischenspeicherung. Über die beiden Speicher A und B liegen die Daten an der ALU oder an dem Ausgangs-Multiplexer an. Das Q-Register ist ein Arbeits-Register mit vorgeschaltetem Schieberegister. Auch hier ist ein serieller Ein/Ausgabe-Betrieb möglich.



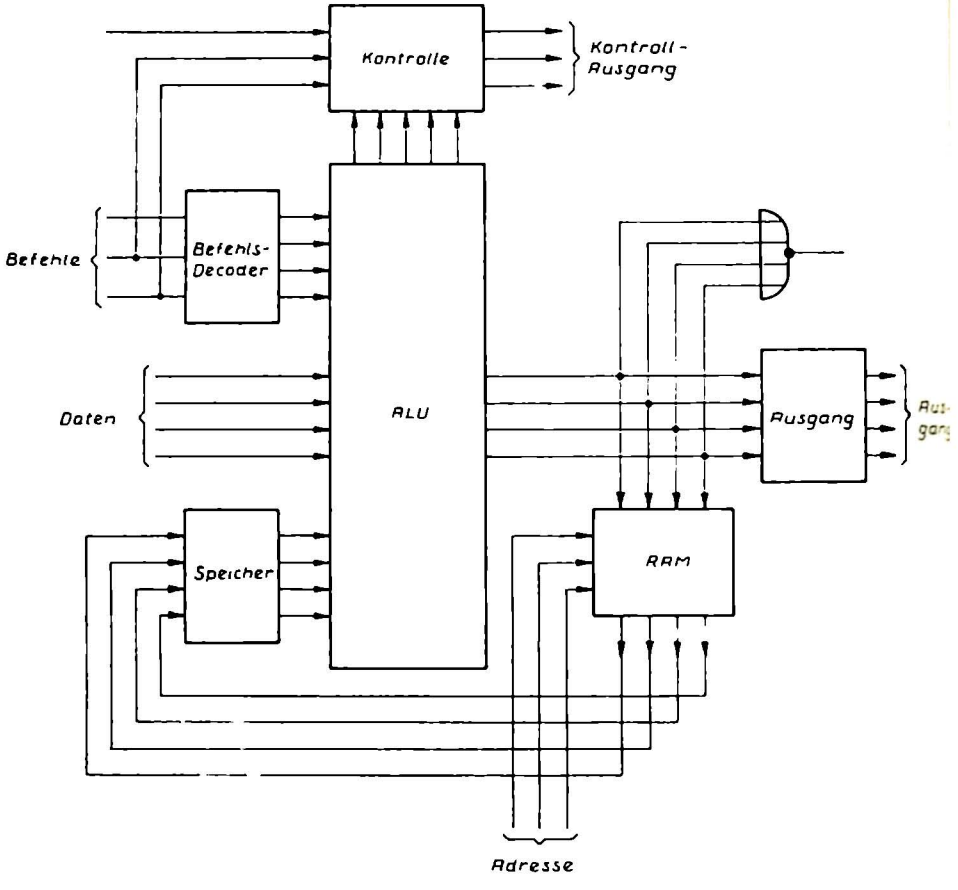
Innenschaltung des Mikroprozessors 2901

Die Firma Advanced Devices stellt den Mikroprozessor 2901 her. Zu diesem Baustein gibt es folgende weitere Bausteine:

Am 2901. 4-Bit-Mikroprozessor-Slice
Am 2901: Übertrags-Einheit zur Rechenkapazitätserweiterung
Am 2905: BUS-Interface-Schaltkreis mit offenem Collektorausgang
Am 2906: BUS Interface-Baustein mit offenem Collektorausgang
Am 2907: BUS-Interface-Baustein mit offenem Collektorausgang
Am 2909: Mikroprogramm-Sequencer
Am 2910: Mikroprogramm-Kontroller
Am 2911: Mikroprogramm-Sequenzer
Am 2913: Unterbrechungs-Baustein
Am 2914. Unterbrechungs-Baustein-Kontroller
Am 2918: Register-Baustein
Am 29803: Sprung-Register-Kontroller
Am 29811. Befehls-Kontroller

Durch die einzelnen Bausteine ergibt sich ein einfaches Mikroprozessor-System mit allen Vor- und Nachteilen der TTL-Technologie. Die Vorteile liegen in der sehr hohen Rechenfrequenz von 15MHz. In Verbindung mit einem Mikroprogramm-Sequenzer kann der Mikroprozessor-Baustein auch mit Einzeltakt (Gleichstrom-Betrieb) angesteuert werden. Der Nachteil ist der hohe Leistungsverbrauch gegenüber der MOS- und der CMOS-Technologie.

Die ALU in dem Baustein 2901 kann acht Funktionen ausführen. Hierzu gehören die logischen und arithmetischen Funktionen. In Verbindung mit dem Schieberegister und anderer Einheiten ergeben sich aber bis zu 202 Funktionsmöglichkeiten.
Die Informationen für den Baustein 2901 liegen direkt an dem Eingang "Daten". Über den Multiplexer gelangen die Daten in die ALU. Der Ausgang der ALU ist über einen weiteren Multiplexer mit dem Ausgang verbunden. Die Ausgangs-Daten der ALU können aber auch direkt in das Q-Register eingeschrieben werden. Das Q-Register ist ein reines Arbeits-Register zum Zwischenspeichern von Informationen. Der Ausgang der ALU ist auch noch mit dem Schieberegister verbunden. Das Schieberegister kann die parallelen Daten seriell auslesen und gleichzeitig seriell Daten einschreiben. Der Ausgang des Schieberegisters ist mit einem Zweifach-Schreib-Lese-Speicher verbunden. Durch die Adressen A und B werden getrennt die Adressen für die einzelnen Speicherplätze bestimmt. Die gespeicherten Informationen in dem RAM werden über die beiden nachgeschalteten Register entweder auf die ALU oder auf den Ausgang geschaltet. Dies ist eine Frage der Programmierung, also der Software oder eine Frage der externen Ansteuerung, also durch die Hardware bedingt.



Innenschaltung des Macrologic-4-Bit-Mikroprozessors

Die beiden Firmen Fairchild Camera and Instruments und Signetics stellen einen 4-Bit-Mikroprozessor in CMOS und in STTL her. Dadurch ergeben sich:

9405:	4-Bit-Mikroprozessor in STTL-Technik
34705.	4-Bit-Mikroprozessor in CMOS-Technik

Beide Typen sind identisch.

Die Architektur der beiden Mikroprozessoren ist einfach. In der Mitte befindet sich eine 8-Funktionen ALU, die ihre Daten direkt und uber einen Speicher erhalt. Der Ausgang der ALU ist uber ein RAM mit dem Eingangs-Speicher gekoppelt. Dadurch können verarbeitende Daten mit den Eingangs-Daten wieder logisch oder arithmetisch verknupft werden. Dies ist von dem Befehlswort an den drei Eingangen abhangig. Der Befehls-Decoder steuert direkt die ALU an.

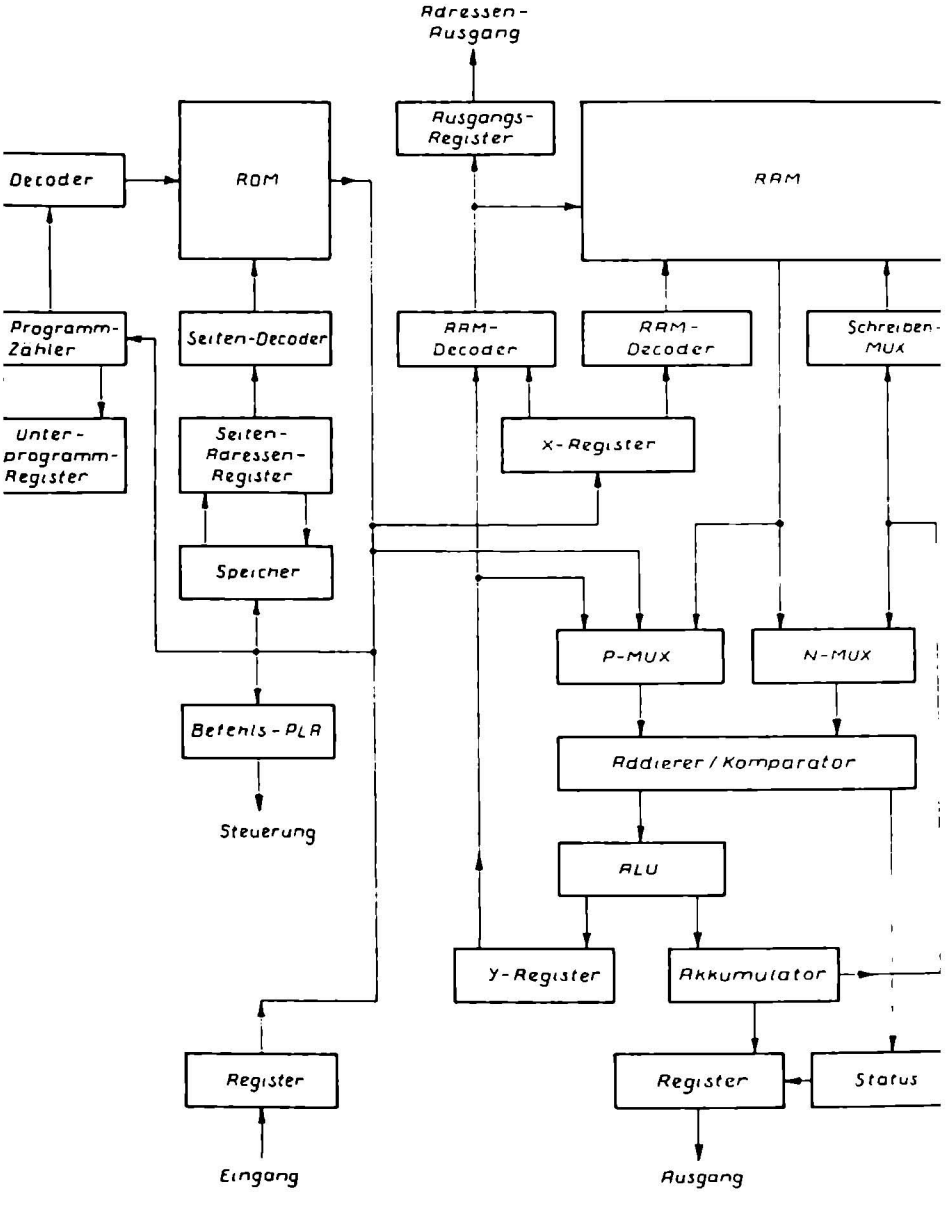
Wichtig fur den Mikroprozessor-Betrieb sind die Zustands-Ausgange. Die Zustands-Ausgange sind unter dem Begriff "flags" zusammengefaßt und steuern direkt die Kontrolle an. Aus dem Funktionsblock in Verbindung mit den drei Kontroll-Eingangen ergibt sich der Kontroll-Ausgang. Die Kontroll-Ausgange können von einer externen Logik verarbeitet werden. Durch eine entsprechende Ruckkopplung und Verarbeitung ergibt sich der Mikroprozessor-Betrieb.

In Verbindung mit dem Schottky-TTL-Mikroprozessor sind folgende Bausteine erhaltlich:

9401:	Check-Generator
9403:	16 x 4 - FIFO
9404:	Daten-Schalter
9405:	Mikroprozessor
9406.	Programm-Stack oder Kellerspeicher
9407:	Daten-Register
9408:	Mikroprogramm-Sequenzer
9410:	16 x 4 -Daten-Register
9423:	16 x 4 - FIFO

Hierzu können noch sämtliche TTL-Bausteine verwendet werden. Durch diese TTL-Technologie ergibt sich ein Rechenfrequenz-Bereich zwischen einem reinen Gleichstrombetrieb (Einzeltakt) und 12MHz.

Die Ausgangs-Daten der ALU können entweder auf den Ausgang gegeben oder in dem RAM (Schreib-Lese-Speicher) zwischengespeichert werden. Die Adressen für das interne RAM liegen extern an. Dadurch lassen sich bis zu acht Informationen in dem RAM speichern. Bei einer externen Adressierung liegt die gespeicherte Information an dem Speicher an und wird mit einem Taktimpuls gespeichert. Diese Art der Speicherung ist für Rechenvorgänge wichtig.



Innenschaltung der TMS-1000-Familie

Die Firma Texas Instruments stellt die TMS1000-Mikroprozessor-Familie in NMOS oder PMOS her. Die Firma Motorola stellt dazu die CMOS-Version her.

Die TMS1000 - Serie beinhaltet auf dem Chip ein ROM mit 16384 Speicherplatzen, ein RAM mit 512 Speicherplätzen und einer ALU mit 54 Befehlen. Dadurch ergibt sich ein sehr komplexer Anwendungsbereich.

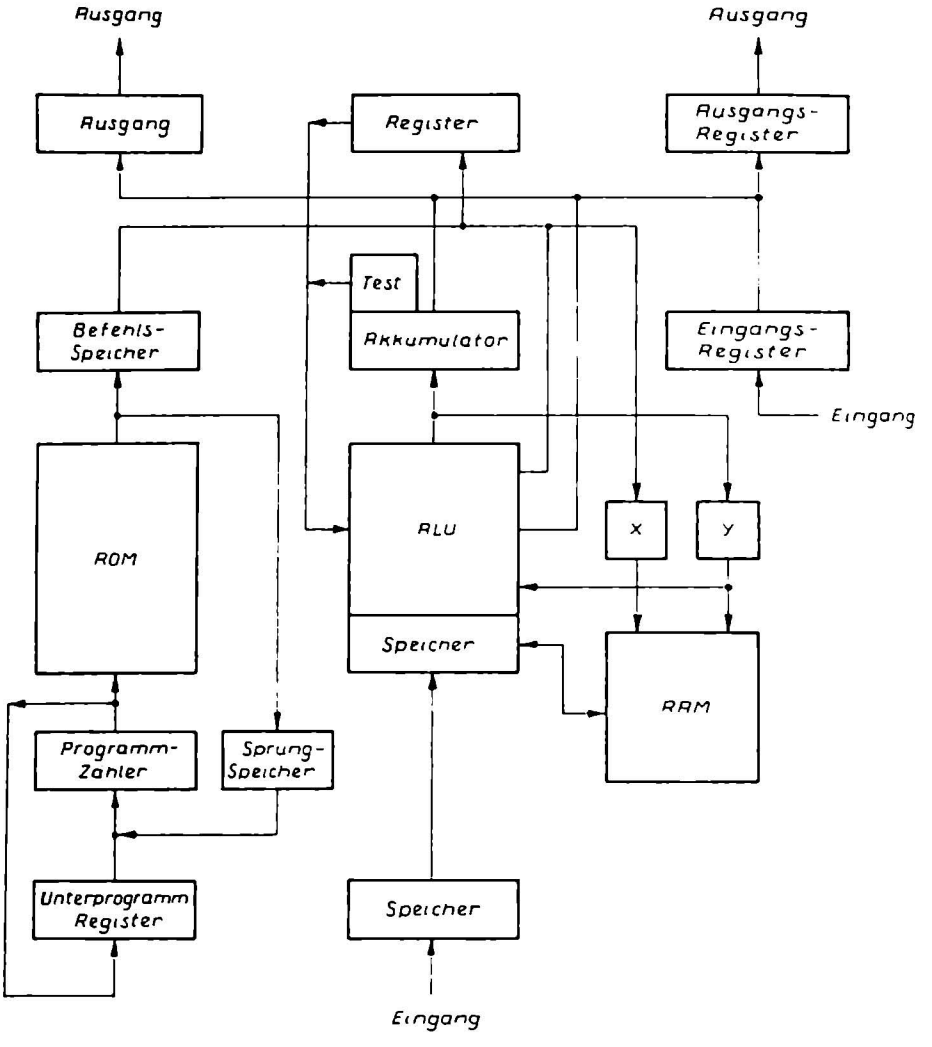
Der Aufbau der TMS1000-Familie erscheint im ersten Moment etwas kompliziert, da zahlreiche Register enthalten sind. Über den Eingang erhalt der Mikroprozessor seine Daten und Befehle. Die Daten konnen in dem RAM abgespeichert werden und so später der ALU zugangig sein. Die Befehle können ebenfalls in dem RAM abgespeichert werden. Bei den Adressen gilt die gleiche Bedingung. Dies gilt besonders fur Sprung-Adressen.

Im üblichen Mikroprozessor-Programmablauf steuert der Programmzahler den Ablauf eines Programmes. Der Programmzähler steuert den Decoder und der seinerseits das ROM (Festwertspeicher) an. In Verbindung mit dem Seiten-Decoder ergeben sich 16 verschiedene Seiten mit je 256 Möglichkeiten zu 4 Bits. Durch die Verwendung von 8-Bit ergeben sich noch 128 Möglichkeiten. Dies ist von dem Stand des Programmzählers und des Seiten-Adressen-Registers abhängig. Mit dem Seiten-Adressen-Register wird die Seite und mit dem Programmzähler die Adresse in der Seite bestimmt.

Durch bestimmte Befehle wird das RAM adressiert. Die beiden eingeschriebenen Informationen liegen so an dem P- und N-Multiplexer an und können von dem Addierer/Komparator in Verbindung mit der ALU verarbeitet werden. Der Ausgang des Komparators ist mit dem Status-Register verbunden. Hier werden der Überlauf, Vorzeichen, Parität usw. gespeichert. Der Ausgang der ALU ist mit dem Y-Register oder dem Akkumulator verbunden. Bei einem Ergebnis gibt die ALU die Informationen in das Y-Register. Das Y-Register überträgt die Informationen weiter in den P-MUX oder in den RAM Decoder. Ist das Ergebnis in der ALU für die externe Verarbeitung fertig, so gibt sie der Akkumulator an das Ausgangs-Register weiter.

Der Akkumulator stellt einen einfachen Arbeitsspeicher zum Zwischenspeichern von Ergebnissen dar. Die Daten des Akkumulators liegen an dem N-Multiplexer und an dem Schreiben-Multiplexer. Der Programmierer hat die Möglichkeit, die Daten entweder in das RAM einzuschreiben oder in die ALU zurückzugeben. Bei einer Einschreibung in das RAM bestimmt das X-Register in Verbindung mit dem Y-Register die Adresse.

Fur die externe Steuerung ist das Befehls-PLA zuständig Das PLA ist eine programmierbare Logikanordnung mit 28 Ausgängen. In Verbindung mit den Eingängen und den Adressen können daher komplizierte Ablaufsteuerungen realisiert werden.



Innenschaltung eines PKW-Mikroprozessors

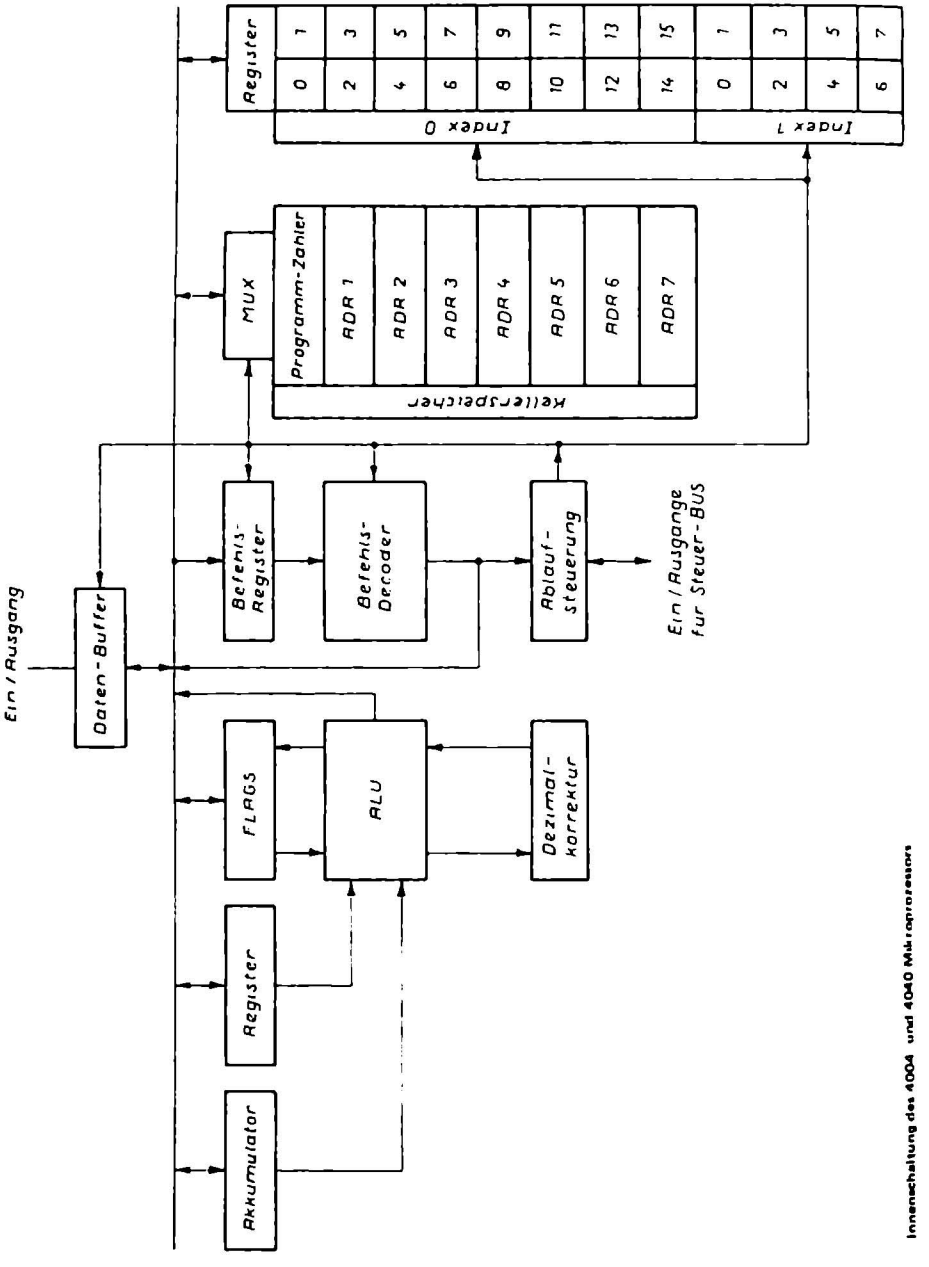
4-Bit-Mikroprozessor in PMOS-Technologie

Die Firma Essex Group Wire Assembly Division stellt einen 4-Bit-Mikroprozessor in PMOS - Technologie her. Der Vorteil bei diesem Baustein ist der Betriebsspannungsbereich zwischen 10V und 18,5V. Dadurch ist dieser Mikroprozessor für den PKW-Betrieb geeignet

Das ROM (Festwertspeicher) enthält die Befehle des Mikroprozessor-Bausteines. Die Adresse der einzelnen Speicherplätze erhält der Festwertspeicher aus dem Programm-Zähler. Der Programmzähler wird von einem externen Takt gesteuert. Die Taktfrequenz liegt zwischen dem reinen Gleichstrombetrieb (Einzeltakt) und 400 kHz. Mit jedem Taktimpuls wird der Zählerstand in dem Programm-Zähler um 1 erhöht
Der Programm-Zähler zählt von 0 bis 1023 und setzt sich dann automatisch auf den Zählerstand 0 zurück. Dieses Zählerprogramm kann durch das Unterprogramm-Register und dem Sprung Speicher beeinflußt werden. Durch einen externen Befehl registriert das Unterprogramm-Register den letzten Zählerstand im Programm-Zähler. Damit wird ein Unterprogramm eingeleitet. Durch den Sprung-Speicher erhält der Mikroprozessor seine Sprung-Adresse. Diese Adresse nimmt der Programm-Zähler auf und setzt dort sein Programm fort. Ist das Unterprogramm beendet, übernimmt der Programm-Zähler den gespeicherten Zustand des Unterprogramm-Registers. Damit kennt der Programm-Zähler wieder seine letzte Adresse vor dem Unterprogramm-Sprung. Er braucht diese Adresse nicht lange zu suchen, da er sie direkt erhält. Der Programm-Zähler ist eine programmierbare Zähleinheit mit 8-Zähler-Bausteinen.

Das ROM decodiert die einzelnen Adressen in Befehle um. Diese Befehle haben eine 8-Bit-Struktur und werden von dem Befehlsspeicher übernommen. Der Ausgang des Befehlsspeichers ist mit dem Register, dem Addierer und dem X-Decodierer des Schreib-Lese-Speichers (RAM) verbunden. Damit erhalten diese Funktionseinheiten ihre Befehle aus dem ROM.

Die Eingangsdaten erhält der Mikroprozessor über den Eingangsspeicher. Hier ist nun die Möglichkeit gegeben, die Informationen direkt in den Addierer einzugeben oder in einem adressierten Speicherplatz abzuspeichern. Dies wird von dem ROM bestimmt. Das RAM hat einen gemeinsamen Ein- und Ausgang. Durch entsprechende Befehle wird abgespeichert oder in dem Addierer die Daten verarbeitet. Am Ausgang des Addierer befindet sich der Akkumulator.
Die ALU kann 41 Befehle ausführen. Diese Befehle sind in dem ROM gespeichert. Die Ergebnisse der ALU können entweder in dem Akkumulator zwischengespeichert oder direkt auf den Ausgang weitergegeben werden. Über das Eingangs-Register können direkt Daten in die ALU eingegeben werden. Dies alles ist eine Frage der Software in Verbindung mit der Hardware.



Innenschaltung des 4004 und 4040 Mikroprozessors

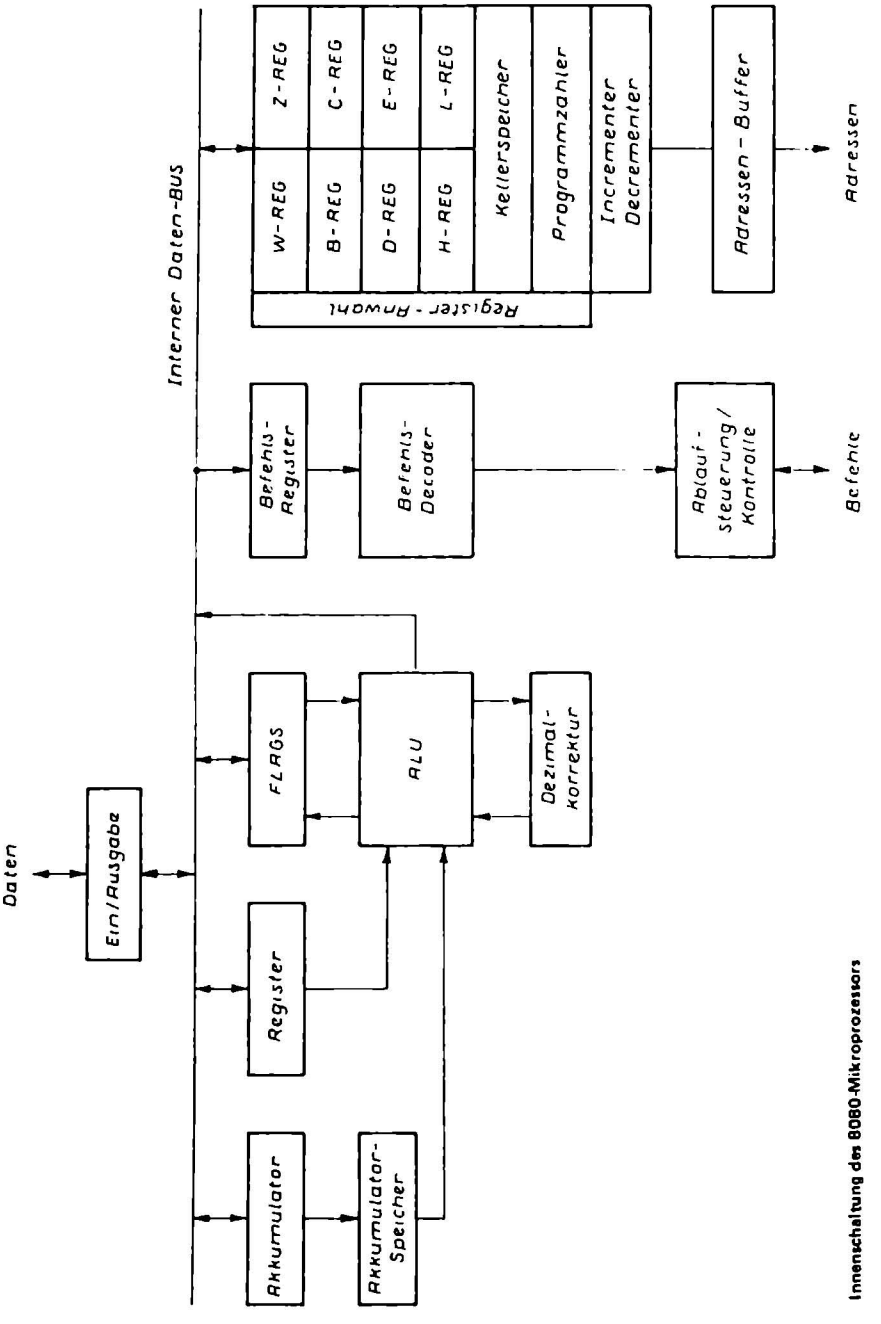
Die Firma Intel stellt den Mikroprozessor 4004 und 4040 her. Es handelt sich um einen 4-Bit-Mikroprozessor mit einem Adressen-BUS von 12-Bits. Dadurch konnen direkt 8192 Adressen adressiert werden. Fur die Befehle ist ein 8-Bit-Wort erforderlich, da bei dem 4004 bis zu 46 Befehle und bei dem 4040 bis zu 60 Befehle ausgefuhrt werden konnen. Für die 4000-Mikroprozessor-Familie gibt es noch folgende Zusatz-Bausteine:

4004:	4-Bit-Mikroprozessor mit 46 Befehlen
4040:	4-Bit-Mikroprozessor mit 60 Befehlen
4003:	10-Bit-Schieberegister
4008:	Speicher-Interface
4002:	320-Bit-Schreib-Lese-Speicher
4001	256 x 8 - ROM
4265:	Programmierbare Ein/Ausgabe-Einheit
4269:	Programmierbare Ein/Ausgabe-Einheit
4201:	Taktgenerator
4289.	Speicher Interface
4308:	1024 x 8 - ROM
4316.	2048 x 8 - ROM
4702:	256 x 8 - EPROM

Die Daten-Eingabe erfolgt bei der Mikroprozessor Familie uber den Daten-Buffer. Damit liegen die Daten an dem internen Daten-BUS. Der interne Daten-BUS ist mit dem Akkumulator, dem Register, den FLAGS, dem Befehls-Register und den beiden Register-Blöcken verbunden.

Der Akkumulator ist ein reines Arbeits-Register. Dadurch kann er Daten fur die ALU zwischenspeichern. Ebenso kann er als kurzer Zwischenspeicher für den internen Daten-BUS verwendet werden. Die ALU erhalt ihre Daten von dem Akkumulator und von dem Register. Die ALU kann logische und arithmetische Befehle durchführen. Die Zustands-Bits werden in der FLAG gespeichert und auf den internen Daten-BUS oder auf die ALU geschaltet. Dadurch sind zahlreiche Funktionen in Abhängigkeit der ALU möglich. Dies ist besonders wichtig für einen bedingten Sprung-Befehl. Zusatzlich befindet sich an der ALU die Dezimal-Korrektur. Tritt eine Pseudotetrade bei einer Rechenoperation auf, so ist der Mikroprozessor in der Lage, diese Pseudotetrade durch einen einfachen Rechenbefehl selbst zu korrigieren.

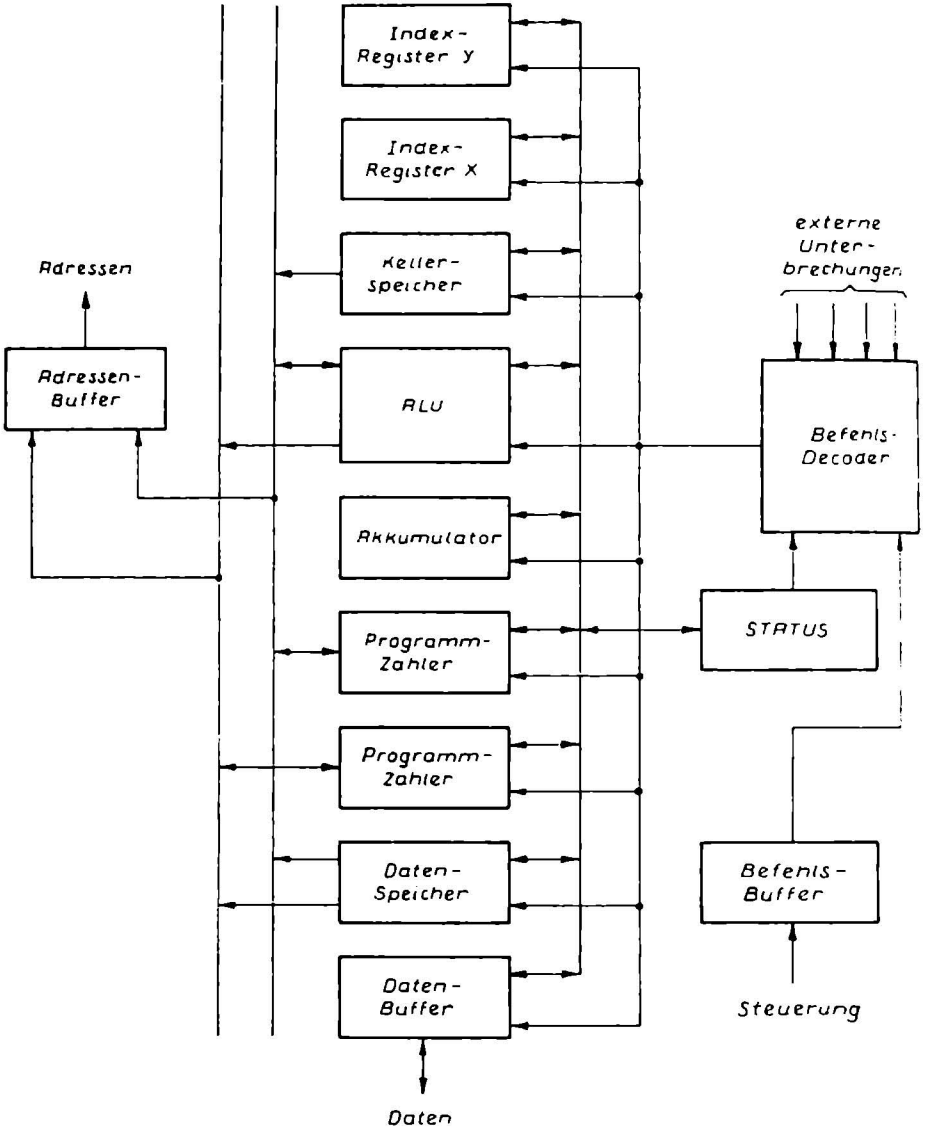
Die Befehle für die 4000-Mikroprozessor-Famile werden über den Daten-BUS eingegeben. Die Befehle werden in dem Befehls-Register gespeichert und an den Befehls-Decoder weitergegeben. Der Befehls-Decoder steuert nun seinerseits den internen Daten-BUS an oder die Ablaufsteuerung fur die externen Bausteine.



Innenschaltung des 8080-Mikroprozessors

Der Mikroprozessor 8080 ist einer der bekanntesten Mikroprozessor-Familien mit zahlreichen Zusatz-Bausteinen:

8080: 8-Bit-Mikroprozessor
8224: Taktgenerator mit Treiber
8228: System-Kontroller mit BUS-Treiber
8238: System-Kontroller und BUS-Treiber
8202: BUS-Treiber
8244: 90-Bit-Eingabe-Codierer
8245: 16-Bit-Eingabe-Codierer
8246: 20-Bit-Eingabe-Codierer
8247: 4-Digit-Anzeigen-Kontroller
8248: 6-Digit-Anzeigen-Kontroller
8253: Programmierbarer Intervall-Zeitgeber
8254: Programmierbares, adressierbares Interface
8255: Programmierbares peripheres Interface
8257: Programmierbarer Speicherzugriff-Baustein
8258: Programmierbarer Speicherzugriff-Kontroller
8259: Programmierbarer Unterbrechungs-Kontroller
8272: Floppy-Disk-Kontroller
8276: CTR-Kontroller
8285: Charakter-Generator
8292: 8-Bit-A/D-Konverter mit 16-Kanal-Multiplexer
8294: Baustein für Digital-Voltmeter mit BCD-Ausgang
8298: Interpreter und Debugger für einen Basic-Anschluß
8250: Asynchrones Übertragungselement
8251: Programmierbares Übertragungselement
8252: Programmierbares Übertragungselement
8261: Programmierbares-Übertragungs-Unterelement
8274: Übertragungs-Kontroller
8356: 2048 x 8 - ROM
8154: 128 x 8 - Bit statisches RAM
8364: Masken-ROM, mit E ein EPROM, 8192 x 8 - Bits
8316: Masken-ROM mit 2048 x 8 - Speicherplätzen
8332: Masken-ROM, mit E ein EPROM, 4096 x 8 - Bits
1702: 256 x 8 - EPROM
8704: 512 x 8 - EPROM
2708: 1024 x 8 - EPROM
8101: Statisches 256 x 4 - Bit - RAM mit separatem Ein/Ausgang
8111: Statisches 256 x 4 - Bit - RAM mit getrenntem Ein- und Ausgang
8102: Statisches 1024 x 1 - RAM
2114: Statisches 1024 x 4 - RAM



Innenschaltung des Mikroprozessors MCS 650X und 651X

Mikroprozessor MCS 650X und MCS 651X

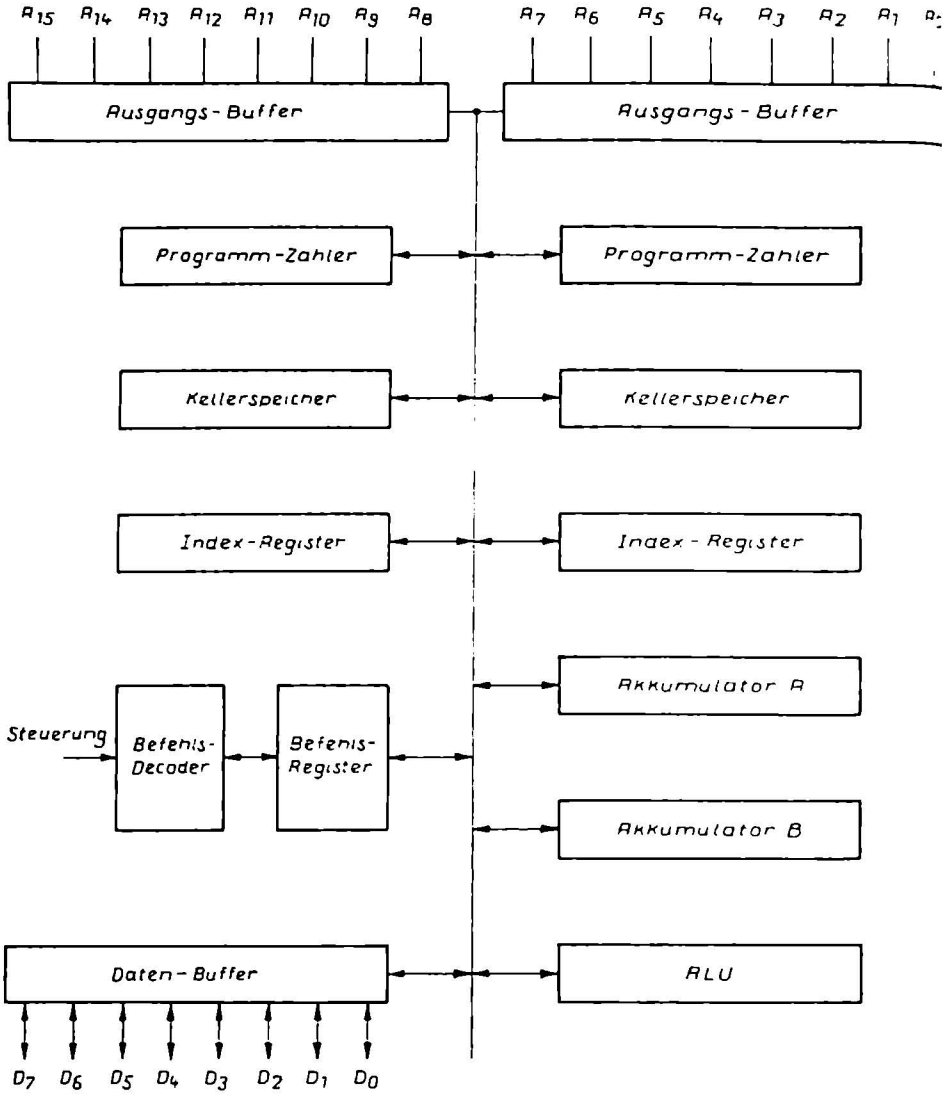
Die Firma MOS-Technology stellt den Mikroprozessor MCS 650X und MCS 651X her. Der Vorteil dieses Mikroprozessors liegt in dem weiten Taktfrequenz-Bereich zwischen 20kHz und 2 MHz.
Von der 65-Familie werden 10 verschiedene Bausteine hergestellt. Die Bausteine unterscheiden sich in den Adressen-Ausgängen und damit im Gehäuse. Es werden folgende Typen geliefert.

6502:	65536 Adressen und 40-Pin-Gehäuse
6503:	4096 Adressen und 28-Pin-Gehäuse
6504:	8192 Adressen und 28-Pin-Gehäuse
6505:	4096 Adressen und 28-Pin-Gehäuse
6506:	4096 Adressen und 28-Pin-Gehäuse
6507:	8192 Adressen und 28-Pin-Gehäuse
6512:	65536 Adressen und 40-Pin-Gehäuse
6513:	4096 Adressen und 28-Pin-Gehäuse
6514:	8192 Adressen und 28-Pin-Gehäuse
6515:	4096 Adressen und 28-Pin-Gehäuse

Der Unterschied zwischen der 0X- und der 1X-Gruppe ist die Taktsteuerung. Die 0X-Gruppe hat einen Taktgenerator direkt auf dem Chip. Bei der 1X Gruppe ist ein externer Takt erforderlich. Bei einem externen Takt wird das System sehr flexibel, da der Takt durch eine externe Logik zusätzlich gesteuert werden kann.
Für diese Mikroprozessor-Familie gibt es noch folgende Bausteine

6520:	Interface-Adapter
6522:	Interface-Adapter mit zwei integrierten Zeitgebern
6530:	16 Ein/Ausgänge mit 8-k-ROM und 64 x 8 -RAM und Zeitgeber
6532:	16 Ein/Ausgänge mit 8-k-ROM und 128 x 8 - RAM und Zeitgeber

Die Architektur dieser Mikroprozessor-Familie zeigt einen internen Daten-, einen Befehls- und Adressen-BUS. Über den Eingangs-Daten-Buffer werden die Daten eingeschrieben oder ausgelesen. Die Daten sind so mit den einzelnen internen Baugruppen verbunden.
Über den Daten-BUS in dem Mikroprozessor werden die einzelnen Baugruppen mit Daten versorgt. Die beiden Index-Register X und Y sind reine Zwischenregister und werden durch einen externen Befehl gesteuert. Darunter ist der Kellerspeicher zum Abspeichern der letzten Adresse vor einem Unterprogramm-Sprung. Die ALU kann 56 Befehle durchführen. Der Akkumulator speichert die Zwischenergebnisse oder die logische Verknüpfung zwischen. Darunter liegt der Programmzähler für das Hauptprogramm und darunter der für das jeweilige Unterprogramm. Der Daten-Speicher kann entweder Daten speichern oder Adressen auf den Adressen-Buffer schalten



Innenschaltung des 6800-Mikroprozessors

Einer der bekanntesten Mikroprozessoren ist der MC 6800 von der Firma Motorola. Hierzu gibt es zahlreiche Bausteine:

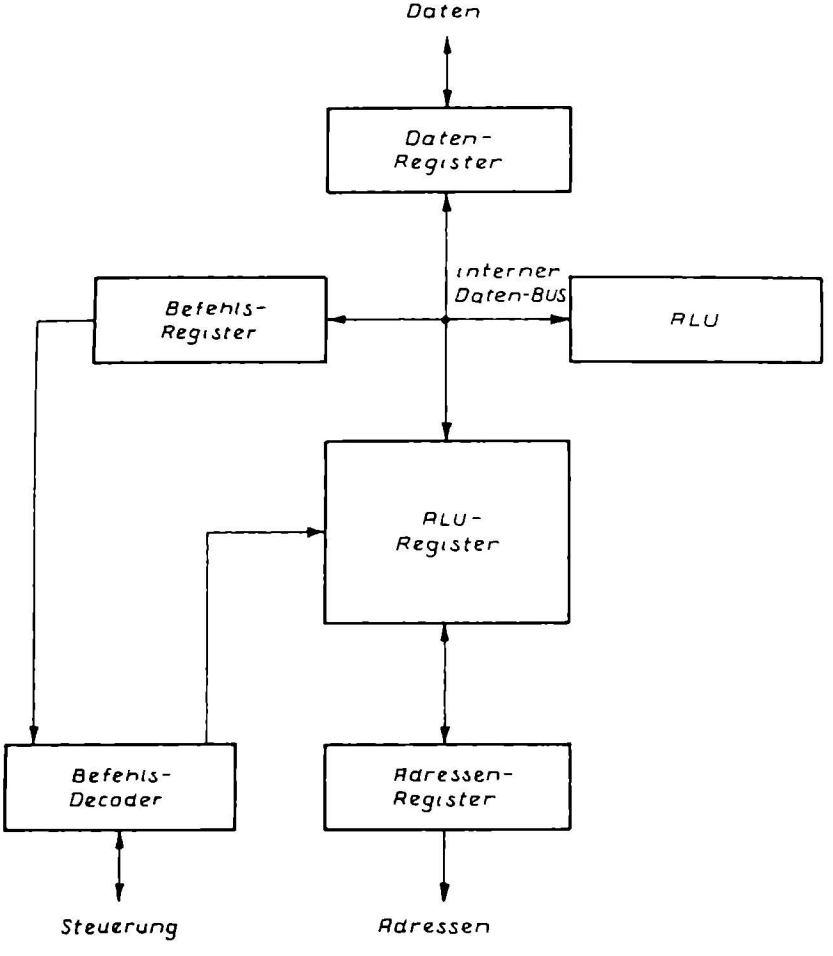
MC 6800:	8-Bit-Mikroprozessor
MC 6820:	Peripherie-Interface-Adapter
MC 6840:	Programmierbarer Zeitgeber
MC 6850:	asynchroner Übertragungsadapter
MC 6854:	Daten-Kontroller
MC 6860:	MODEM-Baustein
MC 6862:	Demodulator
MC 68488	Übertragungs-Interface
MC 6843:	Floppy-Disk-Kontroller
MC 6844:	DMA-Kontroller
MC 6854:	CRT-Kontroller

Die Daten-Ein/Ausgabe erfolgt über den Daten-Buffer. Die Informationen werden entweder in die beiden Akkumulatoren A und B oder in das Index-Register eingeschrieben. Das Index-Register wird für die Zwischenspeicherungen von Daten bei einer Multiplikation benötigt. Die beiden Programmzähler dienen für das Hauptprogramm und das Unterprogramm in Verbindung mit den beiden Kellerspeichern.

Die ALU kann 62 Befehle ausführen. Die Rechengeschwindigkeit wird von einem externen Taktgenerator bestimmt und liegt zwischen einem reinen Gleichstrom-Betrieb (Einzeltakt) und 2 MHz.

Die Adressierung der externen Speicher und Interface-Einheiten wird durch die 16 Adressen-Ausgänge vorgenommen. Durch die 16 Ausgänge ergeben sich 2^{16} Möglichkeiten (65536).

Die Befehle erhält der Mikroprozessor über den Daten-BUS. Die Befehle werden in dem Befehls-Register gespeichert und dem Befehls-Decoder zugeführt. Der Befehls-Decoder gibt seinerseits Steuer-Befehle für die internen und externen Baugruppen ab. Die externen Baugruppen können aber auch den Befehls-Decoder ansteuern. Dies ist besonders mit Programmunterbrechungen interessant. Der Befehls-Decodierer nimmt die externen Steuer-Befehle auf, decodiert sie und gibt sie an das Befehlsregister weiter.



Innenschaltung des Mikroprozessors Z80

Die Firma Zilog stellt den Mikroprozessor Z80 her. Dieser Mikroprozessor hat im wesentlichen die Architektur des 8080. Die Z80 hat die gleiche Befehlsstruktur des 8080, also die 78 Befehle. Bei der Z80 wurde aber der Befehlsvorrat auf 158 Befehle erweitert. Während der Taktfrequenz-Bereich des Mikroprozessors 8080 zwischen 500kHz und 3MHz liegt, liegt der Bereich bei der Z80 zwischen 5kHz und 4,5MHz.

In Verbindung mit dem Mikroprozessor Z80 gibt es folgende Bausteine:

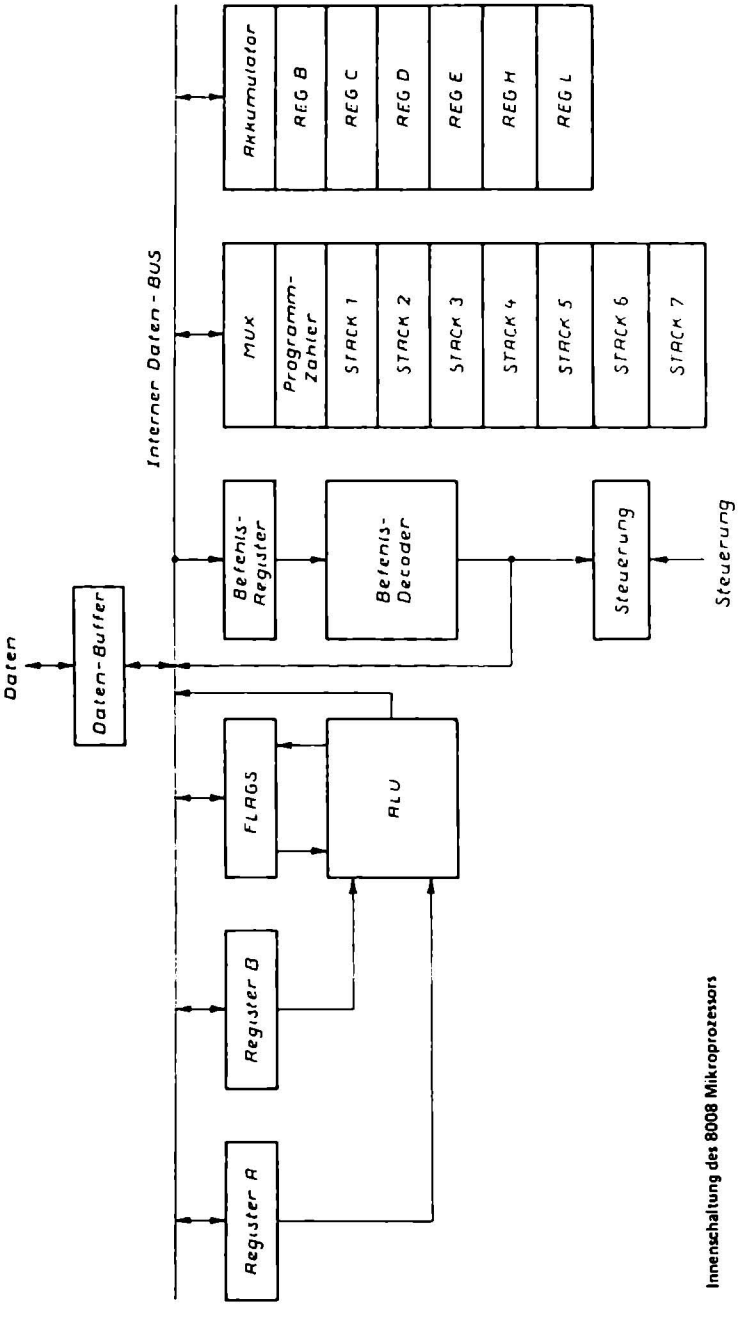
Z80:	8-Bit-Mikroprozessor
Z80-PIO:	Parallele Ein/Ausgabe-Einheit
Z8-CTC:	Zähler- und Zeitgeber-Baustein
Z80-DMA:	Baustein für den direkten Speicherzugriff
Z80-SIO:	Serielle Ein/Ausgabe-Einheit

Für die Z80 ergeben sich 158 Befehle. Diese Befehle teilen sich folgendermaßen auf.

21 Ladekommandos für 8-Bits
20 Ladekommandos für 16-Bits
14 Wechselkommandos für die Register
17 Arithmetische und logische Befehle für 8-Bit-Operationen
11 Arithmetische und logische Befehle für 16-Bit-Operationen
12 Arithmetische Befehle für allgemeine Anwendungen
16 Schiebe- und Rotations-Befehle
11 Sprung-Befehle
9 Setz-, Rücksetz und Test-Befehle
7 Aufruf- und Rückruf-Befehle
12 Ein- und Ausgangs-Befehle

Die Software ist weitgehend mit der Programmiersprache der 8080 identisch. Dadurch kann die Programmänderung in PL/M, PL/Z und BASIC erfolgen.

Die Daten erhält der Z80-Mikroprozessor über das Daten-Register. Das Daten-Register gibt diese Informationen an den internen Daten-BUS weiter. Die Daten gelangen so in die ALU, dem ALU-Register und dem Befehls-Register. Die ALU kann zwei 8-Bit-Rechenoperationen durchführen. Damit ergibt sich ein maximales 9-Bit-Rechenergebnis. Bei einem Multiplikations-Programm mit Stellenverschiebung ergibt sich aus einer zwei 8-Bit-Rechenoperation eine Wortlänge von 16-Bits.
Über den internen Daten-BUS erhält das Befehls-Register die Befehle. Der Befehls-Decoder nimmt die Befehle auf und steuert damit die internen Baugruppen des Mikroprozessors. Durch die Ein- und Ausgabe kann der Befehls-Decoder externe Baugruppen genau so steuern, wie die externen Baugruppen den Mikroprozessor Z80.



Innenschaltung des 8008 Mikroprozessors

Mikroprozessor 8008

Der Vorlaufer dos 8080 ist der 8008. Der Mikroprozessor 8008 von der Firma Intel ist noch in vielen alteren Geräten zu finden. Für neue Geräte soll der Baustein aber nicht mehr verwendet werden.

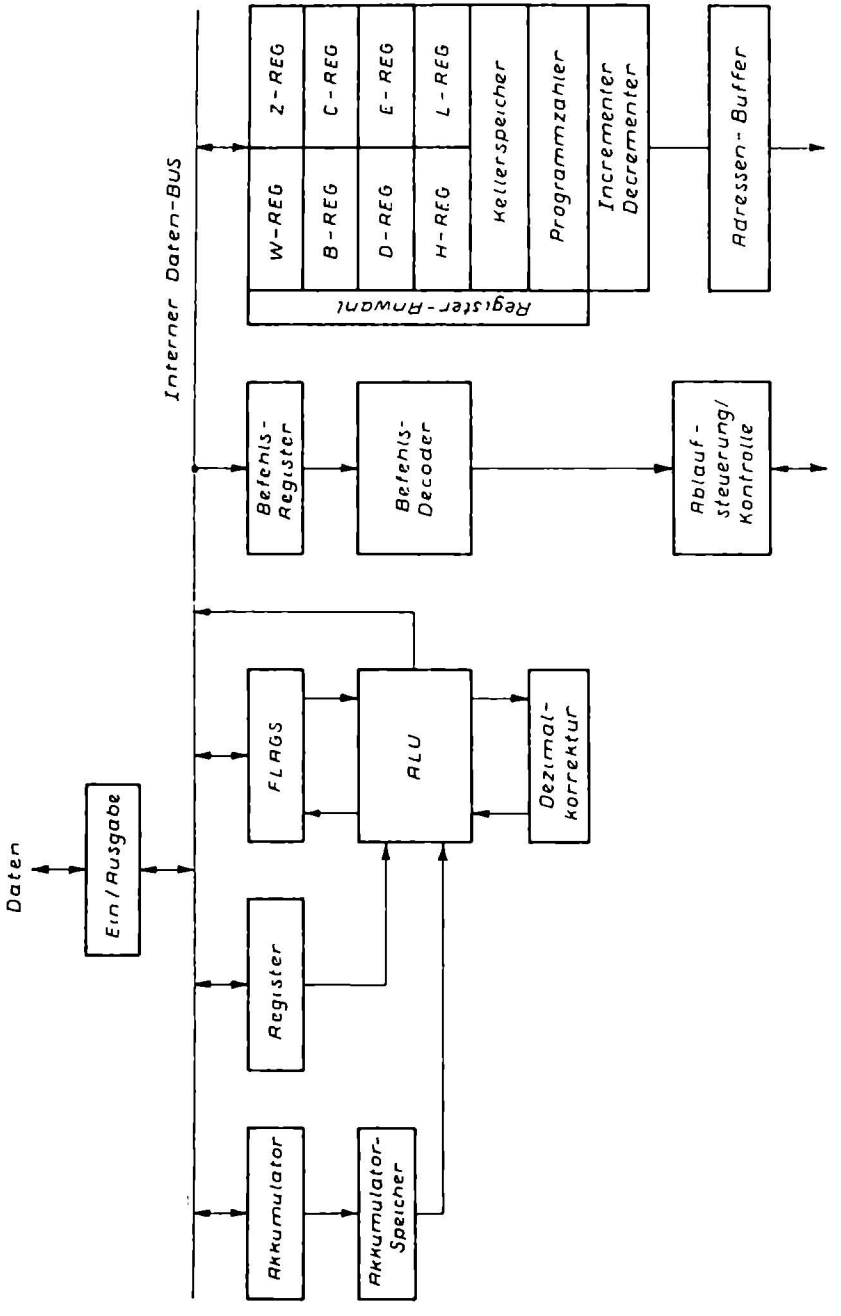
Die Architektur ist die vereinfachte Ausführung des 8080. Der 8008 hat einen Programm-Zahler mit sieben STACK-Registern zum Speichern der Unterprogramm-Adressen. Bei den STACK-Registern handelt es sich um 14-Bit-Wortlangen. Damit liegt der Adressen-BUS mit 2^{14} fest. Lost man 2^{14} auf, ergeben sich 16384 Adressen. Die Ausgabe der Adressen erfolgt im Multiplex-Betrieb, zuerst 8-Bit- der unteren Adresse und dann die 6-Bits der hoheren Adresse. Diese Adressen müssen in einem externen Adressen-Register gespeichert werden, bevor sie die externen Baugruppen adressieren können.

Der Mikroprozessor 8008 erhalt seine Daten und Befehle uber den Daten-Buffer. Die Daten konnen entweder in dem Register A oder B oder in der Register-Anordnung gespeichert werden. Die Register-Anordnung besteht aus sieben Registern, wie dem Akkumulator und sechs allgemeinen Registern.

Die sechs allgemeinen Register können als selbstandige 8-Bit-Register verwendet werden. Es ist keine Registerpaarung wie beim 8080 möglich. 8-Bit-Datenworte können uber den Registerauswahl-Multiplexer zwischen dem internen Daten-BUS und der Register-Anordnung übertragen werden.
Der Programm-Zähler kann von 0 bis 16383 zahlen. Der Programm-Zahler enthalt die Speicheradresse des jeweils gultigen Programmbefehls und wird automatisch bei jedem Befehlsabruf weitergezahlt. Die einzelnen Unterprogramme werden nacheinander in den sieben STACK-Registern gespeichert Damit können diese sieben Register auf einen bestimmten Wert gesetzt werden, so daß die sieben Register nacheinander belegt werden. Die einzelnen STACKs werden durch jeden Befehl heruntergezahlt Der STACK wird jedesmal heruntergezahlt, wenn Adressen in den STACK eingegeben werden und hochgezahlt, wenn Adressen aus dem Stapel entnommen werden.

Die ALU kann bei dem Mikroprozessor 8008 48 Befehle ausfuhren Es ergibt sich folgende Befehlsreihenfolge:

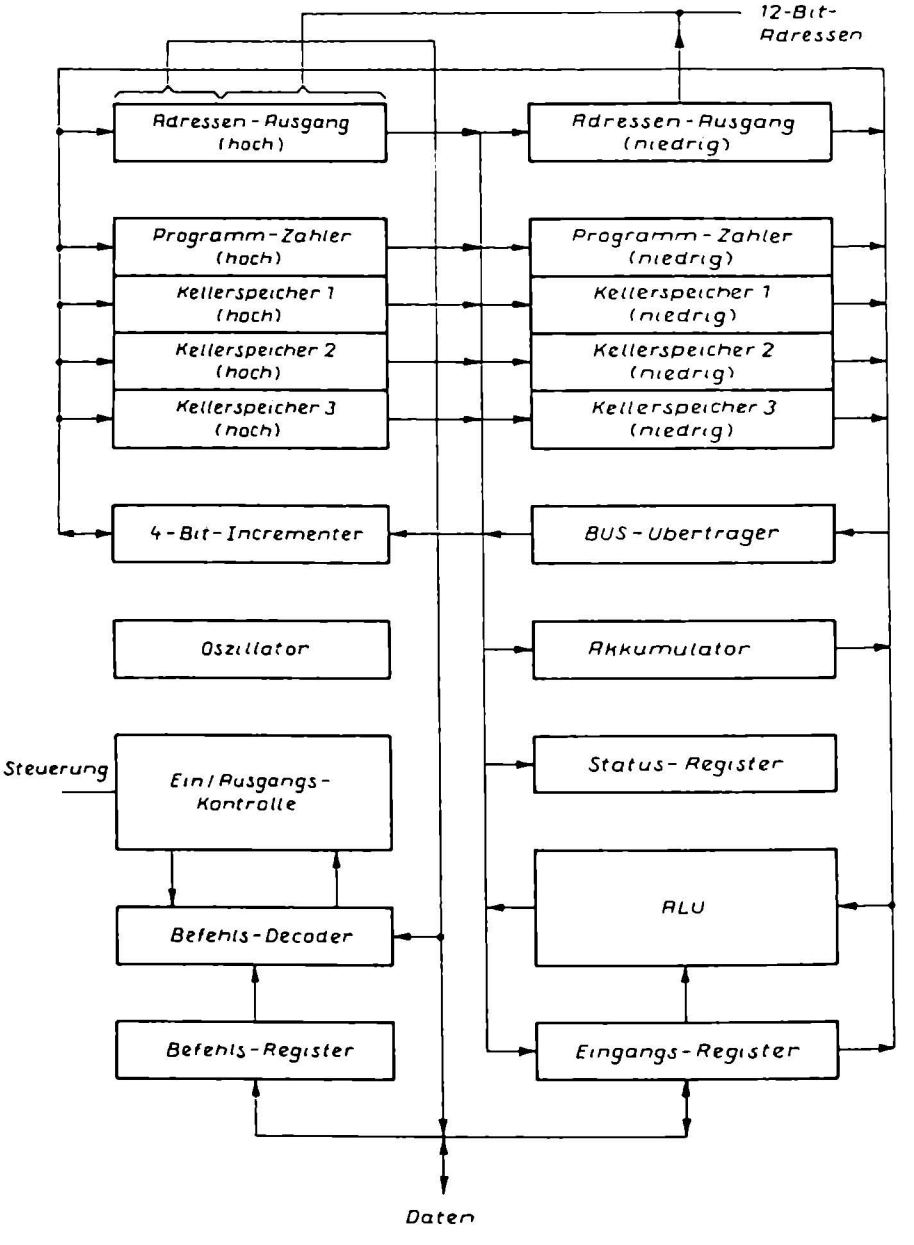
7	Index-Register-Befehle
12	Akkumulatoren-Anweisungen
16	Arithmetische, logische und Schiebebefehle
13	Programm-Zahler-, STACK, Eingabe/Ausgabe und Maschinen-Befehle


Daten
Ein/Ausgabe
Interner Daten-BUS
Akkumulator
Akkumulator-Speicher
Register
FLAGS
ALU
Dezimal-korrektur
Befehls-Register
Befehls-Decoder
Ablauf-steuerung/ Kontrolle
Register-Anwahl
W-REG
B-REG
D-REG
H-REG
Z-REG
C-REG
E-REG
L-REG
Kellerspeicher
Programmzähler
Incrementer Decrementer
Adressen-Buffer

Aus dem 8080 ist der Mikroprozessor 8085 entstanden. Dadurch ist der 8085 Software-kompatibel zum 8080. Während bei dem 8080 die Adressen in 16-Bit-parallel anliegen, hat der 8085 nur 8-Bit-parallele Adressen. Die anderen 8-Bits für die Adressen werden über den Daten-BUS ausgegeben und in einem externen Adressen-Register gespeichert. Aber dafür hat der 8085 fünf verschiedene Möglichkeiten der vektoriellen Programmunterbrechung. Für den 8085 gibt es verschiedene Bauelemente.

8085:	8-Bit-Mikroprozessor
8155:	256 x 8 - Bit-RAM mit integrierten Zeitgebern
8355:	Masken-ROM mit 16 Ein/Ausgangslinien
8755:	EPROM, mit gleicher Pinbelegung wie 8355
8205:	1-aus 8-Decoder
8121:	8-Bit-Register für kombinierten Ein/Ausgabe-Betrieb
8214:	Prioritäts-Unterbrechungsprogramm-Baustein
8216:	Zweiweg-BUS-Treiber
8226:	Invertierender Zweiweg-BUS-Treiber
8253:	Programmierbarer Zeitgeber
8259:	Unterbrechungs-Kontroller
8279:	Anzeige-Interface-Baustein

Die Daten-Eingabe oder die Daten-Ausgabe erfolgt über die Ein/Ausgabe. Der interne Daten-BUS verbindet den Akkumulator, das Register, die FLAGS, die ALU, das Befehls-Register und die Register-Anordnung miteinander. Die Register-Anordnung besteht aus acht Register-Teilen, dem Kellerspeicher für die Speicherung der letzten Adresse bei einem Unterprogrammsprung, dem Programmzähler, dem Incrementer (Erhöher) und dem Decrementer (Verringerer).
Der Programmzähler oder Befehlszähler enthält die Speicheradresse des jeweils gültigen Programmbefehls und wird automatisch bei jedem Befehlsabruf um eine Eins weitergezählt. Der Kellerspeicher (Stapelspeicher oder Stack-Pointer) enthält die Adressen der nächsten verfügbaren Stapelstelle im Speicher. Der Kellerspeicher kann per Programm auf einen Anfangswert gesetzt werden, so daß er einen beliebigen Bereich des externen Schreib-Lese-Speichers als Stapel für die letzte Adresse verwenden kann. Der Kellerspeicher wird jedesmal heruntergezählt,wenn Daten in den Speicher eingegeben werden und jedesmal hochgezählt, wenn Daten aus dem Speicher entnommen werden.
Die sechs allgemeinen Register aus der Register-Anordnung können entweder als selbständige 8-Bit-Register oder auch als 16-Bit-Registerpaare verwendet werden. Die Zwischenspeicher-Registerpaare W und Z sind nicht durch das Programm adressierbar und werden für die interne Ausführung von Befehlen benützt.



Innenschaltung des Mikroprozessors SC/MP

8 Bit-Mikroprozessor SC/MP

Die Firma National Semiconductor stellt folgende Mikroprozessoren her:

SC/MP: 8-Bit-Mikroprozessor in PMOS
SC/MPII: 8 Bit-Mikroprozessor in NMOS

Beide Mikroprozessoren sind sich im Aufbau völlig identisch. Die NMOS-Technologie bringt gegenüber der PMOS-Technologie eine schnellere Zugriffszeit mit sich. Bei der NMOS-Technik betragt die Zugriffszeit für einen Befehl von 5µs, gegenüber von 13µs in PMOS. Dadurch können auch andere Taktfrequenzen verwendet werden, wie die nachfolgende Tabelle zeigt.

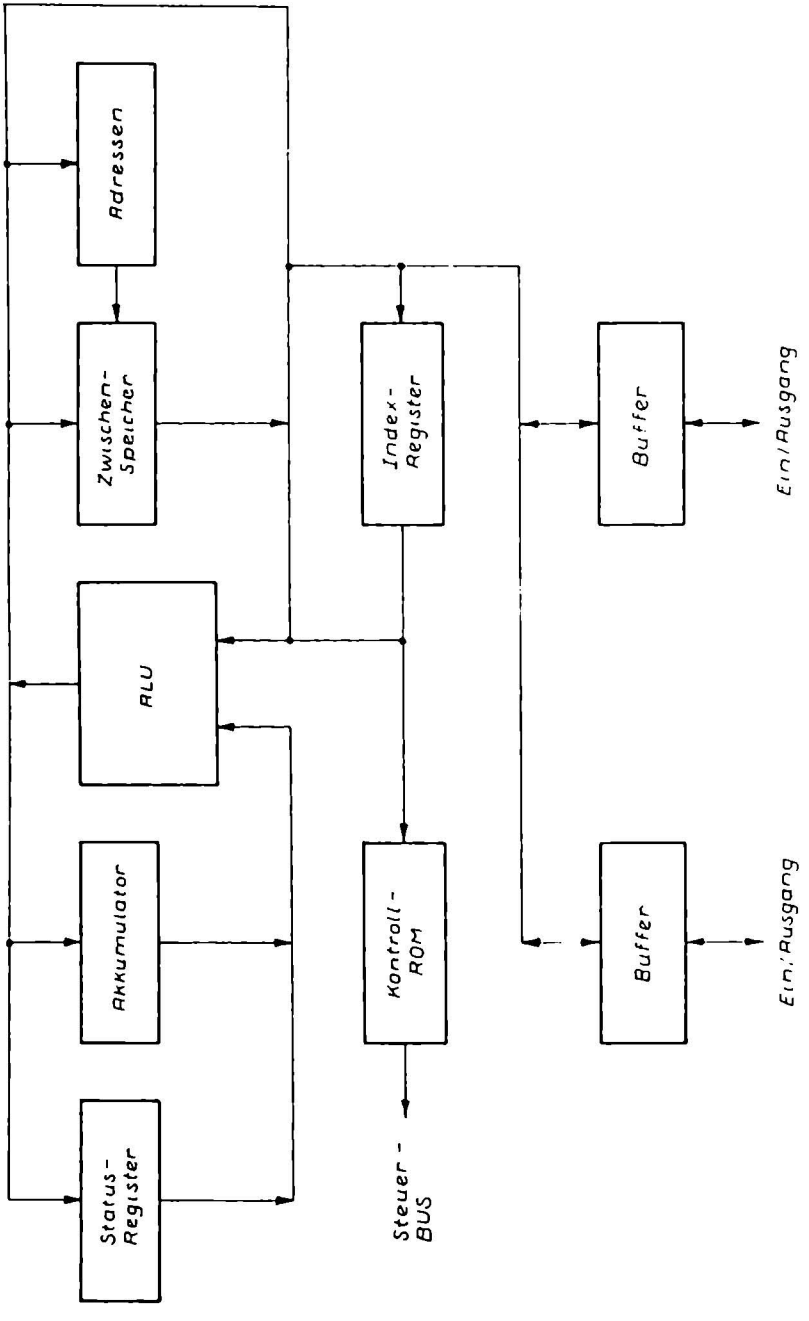
SC/MP: DC bis 1MHz
SC/MPII: DC bis 4MHz

Beide Mikroprozessoren sind sehr einfach im Aufbau und in der Programmierung. Die Datenlange betragt 8-Bits und der Mikroprozessor kann dabei 65536 Adressen ausgeben. Die Adressen werden über einen 12-Bit-Adressen-BUS ausgegeben. Die vier werthöchsten Adressen sind auf dem Daten-BUS zu finden. Damit ist ein externer Adressen-Speicher notwendig. Die 12 niederwertigsten Adressen liegen an dem externen Adressen-Speicher an. Durch den Adressierungs-Befehl auf dem Daten-BUS kann bestimmt werden, in welchen Adressen-Speicher die Adressen eingeschrieben werden.
Die ALU des SC/MP und SC/MPII umfaßt 46 Befehle. Die Befehle werden über den Daten-BUS eingegeben. Das Befehls-Register speichert die Befehle und der Befehls-Decoder decodiert sie. Damit wird die Eingang/Ausgangs-Kontrolle und der interne Daten-BUS angesteuert. Der Befehls-Decoder kann auch über die Ein/Ausgangs-Kontrolle von extern gesteuert werden, z.B. bei einem Unterbrechungs-Kommando. Dadurch ist die Programmierung sehr einfach.

In Verbindung mit den beiden Mikroprozessoren gibt es noch folgende Bausteine.

DM 8334: 8-Bit-adressierbarer Speicher
DM 8131: 6-Bit-Komparator
DM 8546: 8-Bit-Schieberegister für Ein/Ausgabe-Betrieb
MM5307: Baud-Rate-Generator und programmierbarer Zeitgeber
DS 8692: Interface-Schaltkreis

Der SC/MP-Mikroprozessor ist auch als Übungs-Programmier-Kit erhältlich. Damit kann eine einfache Einführung in die Mikroprozessor-Technik erreicht werden


Adressen
Zwischen-Speicher
Index-Register
Buffer
Ein/Ausgang
ALU
Akkumulator
Kontroll-ROM
Buffer
Ein/Ausgang
Status-Register
Steuer-BUS

Die Firma Fairchild Semiconductor stellt den Mikroprozessor 3850 her. In Verbindung mit anderen Bauelementen entstand der bekannte Mikroprozessor F8.
Zu dem Mikroprozessor-Baustein werden folgende Bausteine geliefert:

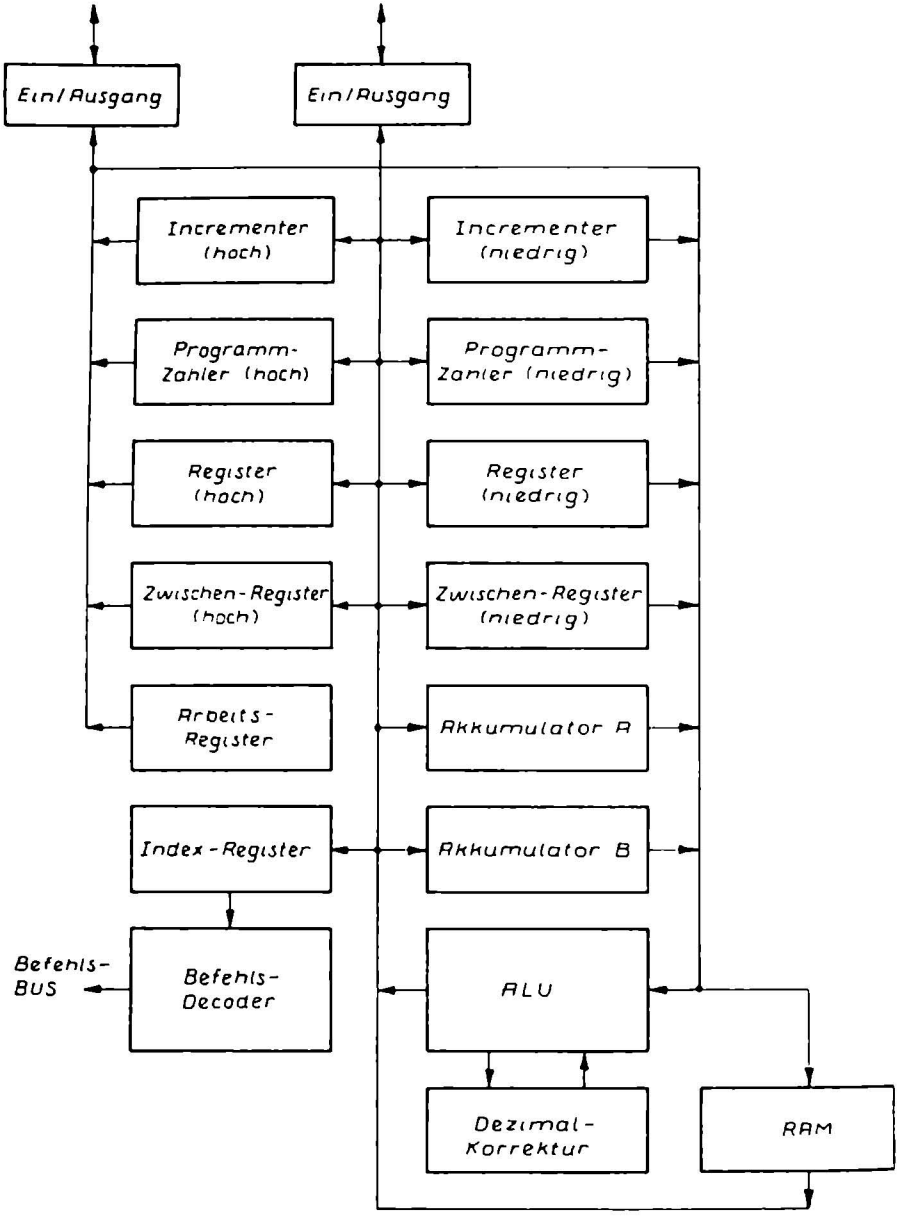
F3850: 8-Bit-Mikroprozessor
F3851: 1-k-ROM mit Zeitgeber
F3852: Dynamisches Speicher-Interface
F3853: Statisches Speicher-Interface
F3854: Direktes-Speicher-Interface
F3856: 2-k-ROM mit Zeitgeber
F3857: 2-k-ROM mit Zeitgeber
F3861: Ein/Ausgabe-Einheit mit Zeitgeber
F3871: Ein/Ausgabe-Einheit mit Zeitgeber

Die Architektur des Mikroprozessors 3850 zeigt eine ALU mit 76 Befehlen, einem 8-Bit-Akkumulator, einem 8-Bit-Zwischenspeicher mit 64 Speicherplätzen und einem Index-Register.
Der Mikroprozessor erhält seine Daten über die beiden Buffer. Die Daten können in den Akkumulator oder in dem Zwischenspeicher abgelegt werden. Bei der Verwendung des Akkumulators muß zuerst ein Befehl (speichere in ACC) kommen und danach das Datenwort. Wird der Zwischenspeicher verwendet, muß zuerst ein Befehl (speichere in TR) dann die Adresse und zum Schluß das Datenwort kommen. Es ist ein 3-Byte-Befehl notwendig.

In dem Zwischen-Register konnen bis zu 64-Datenwörter mit 8-Bit zwischengespeichert werden. Durch einen 2-Byte-Befehl über den Buffer an das Kontroll-ROM wird zuerst der Befehl eingegeben und danach die Adresse. Das Datenwort wird ausgelesen und in dem Akkumulator zwischengespeichert. Über die ALU mit ihren 76 Funktionen können dann die arithmetischen oder logischen Funktionen durchgeführt werden.
Die 76 Befehle der ALU setzen sich folgendermaßen zusammen:

16 Akkumulatoren-Befehle
8 Speicher-Befehle
12 Sprung-Befehle
13 Adressierungs-Befehle
15 Befehle für den Zwischen-Speicher
13 Kontroll-Befehle

Alle diese Befehlsstrukturen sind auch in dem F8-Mikroprozessor enthalten. Die Programmiersprache ist FAIRBUG, eine etwas geänderte Programmiersprache des FORTRAN IV.



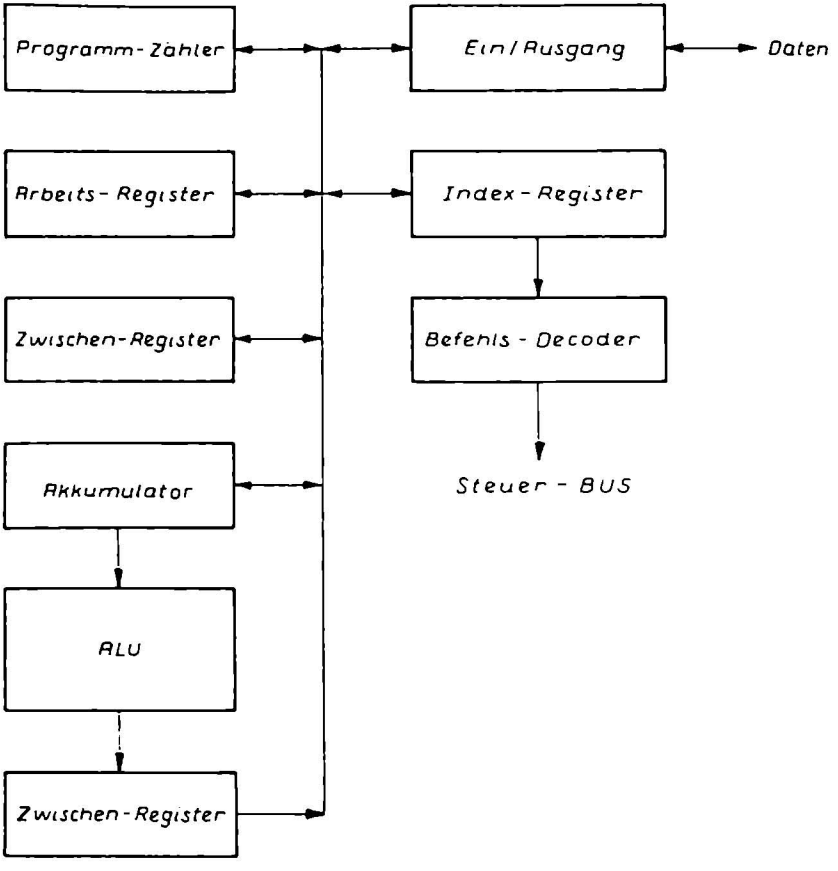
Innenschaltung des Mikroprozessors MC 6802

Der Mikroprozessor 6800 wurde erweitert und trägt nun die Bezeichnung 6802. Die Firma Motorola stellt den Typ MC6802 her.
Der Mikroprozessor MC6802 hat einen internen Oszillator, der von einem externen Quarz angesteuert wird und einen Schreib-Lese-Speicher mit 128 Speicherplätzen, der eine Bitlänge von 8 aufweist. Zuerst muß an diesem RAM eine Adresse durch den internen Daten-BUS anstehen. Die Adresse wird gespeichert und adressiert den Speicherplatz. Danach erhält der adressierte Speicherplatz über den internen Daten-BUS die Information zum Einschreiben. Das Auslesen einer Speicherzelle ist etwas einfacher. Ist die Speicherzelle adressiert, liegt die Information an dem internen Daten-BUS sofort an.

Für den Mikroprozessor MC6802 gibt es noch folgende Bausteine

MC6846:	ROM mit Zeitgeber
MC6820:	Interface-Adapter
MC6840:	Programmierbarer Zeitgeber
MC6850:	Asynchroner Übertragungs-Adapter
MC6854:	Daten-Kontroller
MC6860:	MODEM
MC6862:	Demodulator
MC68488:	Interface-Adapter für allgemeine Anwendungen
MC6843:	Floppy-Disk-Kontroller
MC6844:	DMA-Kontroller
MC6845:	CRT-Kontroller

Die Datenwortlänge beträgt 8-Bits. Durch die 16 Ausgänge der Adressen können bis zu 65536 einzelne Adressen angesteuert werden. Für die Befehle mit Adressen sind 8, 16 oder 24-Bit erforderlich. Bei einem einfachen Befehl benötigt man 1-Byte. Für einfache Rechenbefehle mit Datenwort einen 2-Byte-Befehl, wobei das erste Byte den Befehl und das zweite Byte das Datenwort enthält. Für einen Speicherbefehl wird ein 3-Byte-Befehl benötigt, wobei das erste Byte den Befehl, das zweite und dritte Byte dann die Adresse enthält.
Die ALU kann 72 Befehle durchführen. Hierzu gehören die arithmetischen Rechenoperationen binär und dezimal. Durch die Dezimal-Korrektur kann der Mikroprozessor MC6802 auch im Dezimal-Code arbeiten. Neben diesen arithmetischen Funktionen können noch logische Verknüpfungen durchgeführt werden. Weitere Befehle sind zum Verschieben von Informationen vorhanden und Sprung-Befehle.
Die Software für den Mikroprozessor kann in BASIC oder FORTRAN durchgeführt werden. Der EXORciser ist hierzu ein komplettes Entwicklungssystem.



Innenschaltung des Mikroprozessors IM6100

Mikroprozessor IM6100

Seit ca. 1968 stellt die Firma Digital Equipment den Mikrocomputer PDP-8 her. Der Mikrocomputer wurde in ca. 5500 000 Einheiten hergestellt und stellt damit einen der weitverbreitesten Mikrocomputer dar.
Die Firma Intersil hat durch ihren IM6100 einen Software kompatiblen Mikroprozessor-Baustein mit folgenden Bausteinen hergestellt:

IM6100.	12-Bit-Mikroprozessor
IM6101:	Programmierbarer Parallel-Interface
IM6102:	Speicher- und DMA-Baustein
IM6103:	Speicher-Baustein
IM6402:	UART-Baustein
IM6403:	UART-Baustein
IM6312:	Masken-ROM
IM6512:	CMOS-RAM
IM6603:	CMOS-EPROM

Die Architektur des IM6100 ist einfach. Die Befehle und die Daten erhält der IM6100 über den Ein/Ausgang. Die Befehle werden in dem Index-Register gespeichert und an den Befehls-Decoder weitergegeben. Die Daten können in dem Arbeits-Register, dem Zwischen-Register oder dem Akkumulator zwischengespeichert werden. Die ALU ist nur über den Akkumulator zu erreichen. Die Ausgabe der ALU-Daten erfolgt über ein weiteres Zwischen-Register.
Die Befehls-Eingabe und die Daten haben eine 12-Bit-Länge. Der Baustein IM6100 kann 80 Befehle ausführen. Durch einen 8-Bit-Ausgang lassen sich 256 Adressen direkt ansteuern. Reicht dieser Bereich nicht aus, können über den Daten-BUS weitere Adressen bestimmt werden.

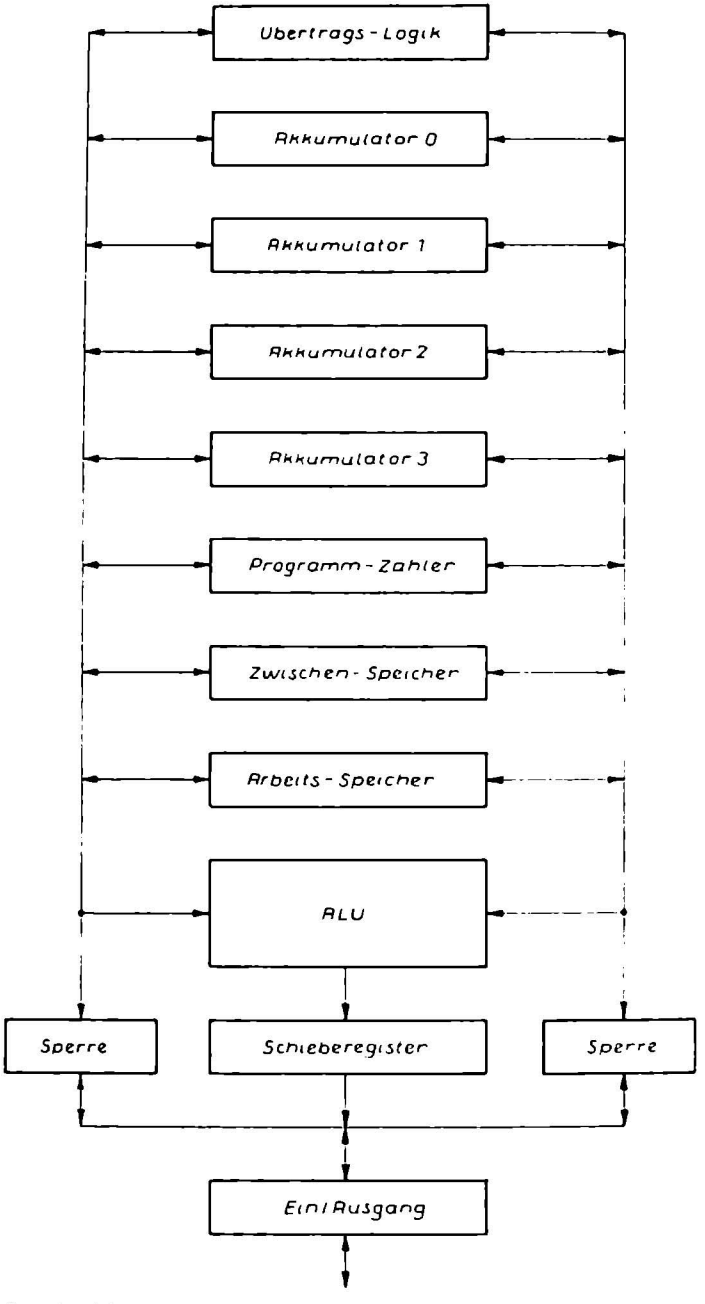
Die Rechenfreqeunz bei dem Baustein IM6100 reicht von dem reinen Gleichstrom-Betrieb (Einzeltakt) bis zu 8MHz. Dabei kann die Betriebsspannung zwischen 5 und 12V schwanken. Es ergibt sich folgende Stromaufnahme:

5V = 2,5mA
10V = 10mA

Die Befehle des IM6100 teilen sich in Speicher-Befehle, Operations-Befehle und Transport-Befehle zwischen den internen und externen Daten-BUS auf.

Die Programmiersprache für den IM6100 ist FOPAL, eine geänderte FORTRAN-Programmiersprache. Es sind durch die Software des IM6100 mit der PDP-8 ca. 5000 Programme erhältlich.

Zur Erlernung der Programmierung ist der Intercept-Kit erhältlich.



Innenschaltung des mN601-Mikroprozessors

Die Firma Data General Corp. stellt einen 16-Bit-Mikroprozessor in NMOS-Technologie her. Zu dieser Familie gehören folgende Bausteine:

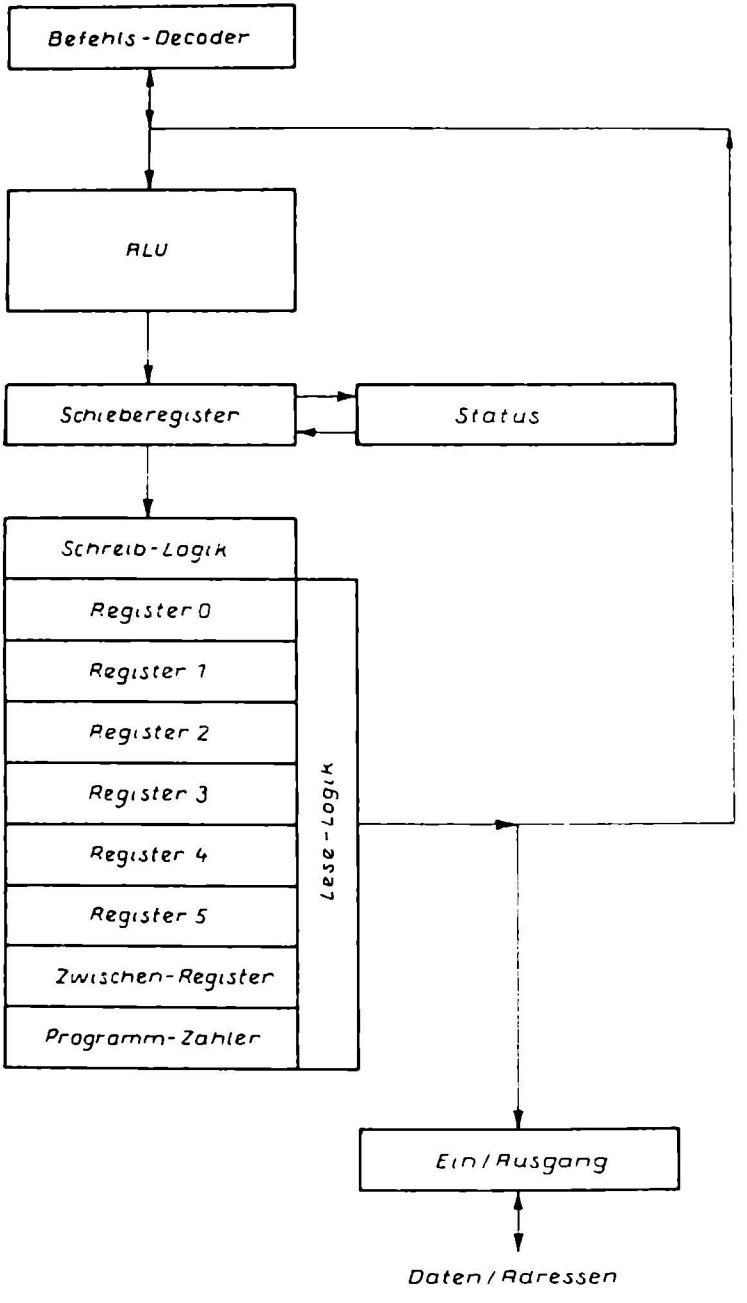
mN601:	16-Bit Mikroprozessor
mN603:	Ein/Ausgabe-Kontroller
mN606:	4-k-dynamisches RAM
mN629:	Sender/Empfanger
mN634:	Sender/Empfanger
mN506:	Lese-Verstärker
mN636:	Sender/Empfanger-Kontroller
mN638:	Takttreiber fur mN606
mN640:	Takttreiber fur mN601

Das Datenwort fur den mN601-Mikroprozessor hat eine Lange von 16-Bit. Die Adressierung für den Mikroprozessor erfolgt über 15-Ausgange, als 15-Bits. Damit konnen 32768 Adressen erreicht werden.
Die Befehle fur den mN601-Mikroprozessor werden über ein 16-Bit-Wort eingegeben. Es können 72 Befehle durchgeführt werden. Die Zugriffszeit der Befehle liegt zwischen 2µs und 60µs. Dies ist von der Befehlsstruktur abhangig.
Der kürzeste Befehl ist eine Addition. Die längste Zeit wird durch einen Divisions-Befehl erreicht.
Der Bereich der Rechengeschwindigkeit liegt zwischen dem reinen Gleichstrombetrieb (Einzeltakt) und 8,4MHz. Für die einzelnen Betriebsspannungen gelten folgende Stromaufnahmen:

5V = 20mA
10V = 30mA
14V = 60mA

Den geringsten Stromverbrauch erreicht man bei 4,25V. Hierbei betragt der Strom nur noch 0,25mA.

Als Programmiersprache kann BASIC oder FORTRAN IV verwendet werden. Damit kann der Mikroprozessor neben den einfachen arithmetischen und logischen Funktionen auch logarithmische, exponentielle, trigonometrische und vektorielle Funktionen durchfuhren.



Innenschaltung des CP 1600 Mikroprozessors

Die Firma General Instrument Corp. stellt die Mikroprozessor-Familie CP1600 her. Zu dieser Familie gehoren folgende Bausteine:

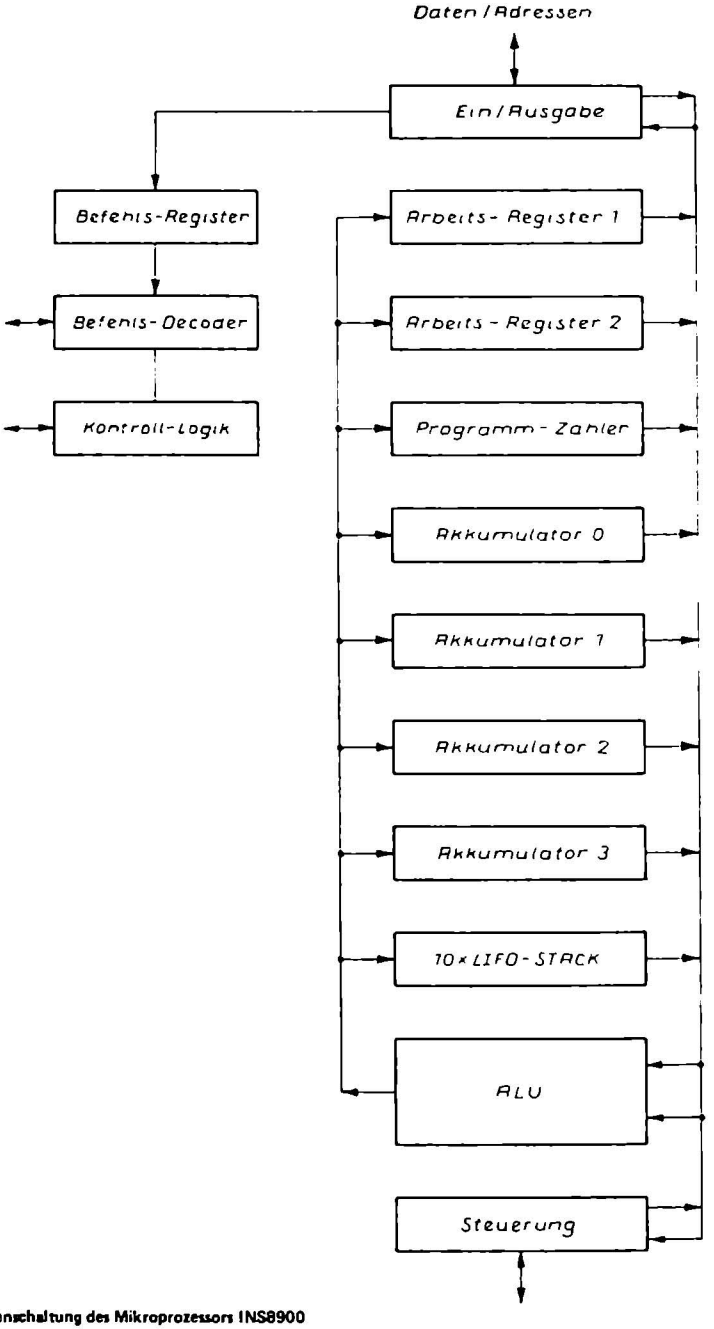
CP1600:	16-Bit-Mikroprozessor bis 3,3MHz
CP1600A:	16-Bit-Mikroprozessor bis 5MHz
CP1610:	16-Bit-Mikroprozessor bis 2MHz
DAC1610:	Zweifacher DAU (Digital-Analog-Umsetzer)
IOB1610:	Ein/Ausgang
MUX1610.	18-Kanal-Analog-Multiplexer
RO-3-5120:	512 x 10-Bit-ROM
RO-3-20480:	2048 x 10-Bit-ROM

Die Daten und die Adressen für den Mikroprozessor CP1600 und CP1610 werden uber den Ein/Ausgang eingeschrieben oder ausgelesen. Dadurch müssen externe Register diese Daten und Adressen aufnehmen und speichern.
Die Ein- und Ausgabe der Daten und Adressen erfolgt 16-Bitweise. Dadurch kann entweder ein 16-Bit-Datenwort oder ein 16-Bit-Adressenwort übertragen werden. Dies ergibt bei dem 16-Bit-Adressenwort einen direkten Adressenbereich von 62236 Adressen.
Der Mikroprozessor CP1600 kann 87 Befehle verarbeiten. Die Taktfrequnez reicht vom reinen Gleichstrombetrieb (Einzeltakt) bis zu 5MHz. Für eine Kontrollfunktion ergibt sich eine Zugriffszeit von 1,6µs und für einen Befehl 4,8µs.

Der Spannungsbereich des Mikroprozessors reicht von 5V bis 12V. Dabei ergibt sich folgende Stromaufnahme.

5V = 12mA
12V = 70mA

In Verbindung mit den externen Baugruppen erfolgt die Programmierung nach der Programmiersprache FORTRAN IV. Als Übungskit wurde hierzu der Gimini-Mikrocomputer entwickelt.



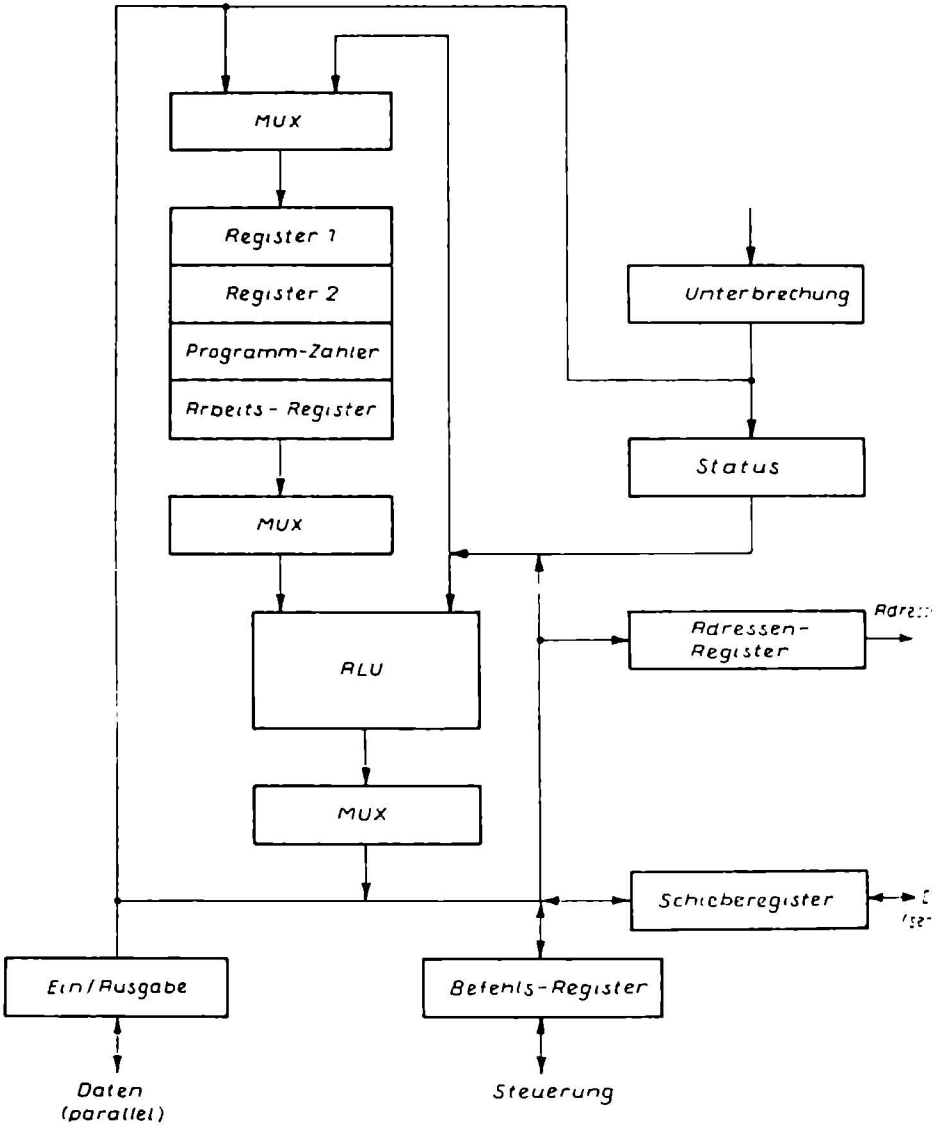
Innenschaltung des Mikroprozessors INS8900

Die Firma National Semiconductor stellt den Mikroprozessor INS8900 her. Über die Ein/Ausgabe werden die Daten und die Befehle seriell in den Mikroprozessor eingegeben. Insgesamt sind durch die Architektur 16 Anschlußpunkte erforderlich.
Die Daten gelangen über den rechten Daten-BUS in die ALU und werden von dort aus in die einzelnen Register übertragen. Der Baustein INS8900 hat hierzu zwei Arbeits-Register und vier Akkumulatoren. Neben diesen beiden Registern gibt es noch einen Programmzähler und einen LIFO.
Der LIFO ist ein spezielles Register, d.h. "last in, first out" Durch diese Struktur der Datenspeicherung ergibt sich ein Verhalten, daß die letzte Adresse als erste wieder ausgelesen wird. Damit können Programmunterbrechungen mit verschiedenen Prioritätsebenen durchgeführt werden. Die letzte Adresse wird durch den Mikroprozessor immer bevorzugt.

Der Adressierungsbereich des Mikroprozessors INS8900 liegt bei 65536 Adressen. Diese Adressen werden von dem Mikroprozessor als Daten behandelt, sofern sie sich auf dem Daten-BUS befinden. Erst wenn sich diese Daten in dem Programm-Zähler oder in einem der zehn STACK-Register befinden, ergibt sich eine Adresse.
Die Ein/Ausgabe des Mikroprozessors ist mit zwei externen Register-Bausteinen verbunden In diesen Register-Bausteinen erfolgt die Zwischenspeicherung. Mit einem Steuer-Befehl kann eine Adresse, ein Datenwort oder eine Adresse erkannt werden.
Die zwei externen Register-Bausteine können jeweils nur zwei Bytes speichern Dadurch ergibt sich für das gesamte Mikroprozessor-System ein kombinierter Daten- und Adressen-BUS. Parallel hierzu liegt dann der Steuer-BUS. In der Technik schaltet man meistens vier Register-Bausteine parallel und steuert sie von der internen Steuer-Logik des INS8900 an. Damit kann der Mikroprozessor seine Adressen von den Daten trennen. Es ergibt sich zwar hierbei ein beträchtlicher Bauteileaufwand, aber die Rechengeschwindigkeit kann erhöht werden.
Der Mikroprozessor INS8900 kann 45 Befehle ausführen. Zu diesen Befehlen gehören folgende Möglichkeiten:

- 12 Arithmetische Befehle und logische Funktionen
- 10 Speicherkommandos
- 13 Zweig- und Sprung-Befehle
- 10 Register-Operationen

Die Rechengeschwindigkeit bei dem Mikroprozessor INS8900 liegt bei 1,5MHz bis 2MHz.
Als Programmiersprache kann eine FORTRAN-CROSS-ASSEMBLER verwendet werden.



Innenschaltung des Mikroprozessors TMS-9900

Mikroprozessor TMS-9900

Die Firma Texas Instruments stellt zwei Mikroprozessor-Typen vor:

TMS 9900:	16-Bit-Mikroprozessor in NMOS-Technik
SPP-9900:	16-Bit-Mikroprozessor in I^2L-Technik

Der Mikroprozessor 9900 hat einen separaten Daten- und Adressen-BUS. Bei dem Daten-BUS konnen uber 16 Punktee die Daten ein- und ausgelesen werden. Die Adressen-Ausgabe erfolgt uber 15 Punkte. Damit ergibt sich ein direkter Addressierungsbereich von 32768 Adressen.
Die TMS-Version der beiden Typen ist in Silicon-Gate-NMOS-Technologie hergestellt. Bei dieser Technologie ergibt sich eine Rechenfrequenz bis zu 3MHz bei einem internen Vier-Phasen-Takt. Die andere Version ist in I^2L-Technologie hergestellt. Dabei ergibt sich eine höchste Rechenfreqeunz von 2MHz.

Die Architektur des Mikroprozessors zeigt zwei Register, einen Programm-Zähler und ein Arbeits-Register. Diese vier Einheiten sind über einen Multiplexer mit der ALU verbunden.
Der Mikroprozessor kann 69 Befehle ausführen. Zu diesen Befehlen gehören 26 arithmetische, logische und manipulierbare Kommandos. Dazu gehören noch 15 interne Register- und Speicher-Befehle, 5 Datentransport Befehle und 2^{24} Kontroll-Funktionen.
Der Baustein hat ein internes Multiplizier-Programm. Durch einen Multiplizier-Befehl werden zwei binare Funktionen in der ALU multipliziert. Hierzu kommen noch 16 Prioritats-Befehle für Programm-Unterbrechungen und einen einfach zu programmierbaren direkten Speicherzugriff. (DMA).

Für die Programmierung dieser Mikroprozessor-Familie ist eine höhere Programmiersprache, die PL/9900, BASIC und FORTRAN erforderlich. Damit können neben den üblichen arithmetischen und logischen Funktionen auch schwierige mathematische und trigonometrische Funktionen durchgeführt werden.

5. Aufbau eines Mikroprozessor-Systems

In Bild 5.1 ist der Aufbau eines Mikroprozessor-Systems gezeigt. Anhand von diesem Bild sollen kurz die wichtigsten Einheiten eines Systems vorgestellt werden. Das System besteht aus folgenden Einheiten:

MPU: Mikroprozessor-Einheit (micro prozessor unit). In dieser Einheit befindet sich das Gehirn eines Systems. Der Baustein selbst besteht aus einer Vielzahl von Funktionsblöcken, den Registerschaltungen, der Arithmetik-Logik-Einheit, der zeitlichen und logischen Steuereinheit, dem Befehls- und Operationsdecodern, den Ein- und Ausgabe-Einheiten und den Akkumulatoren.

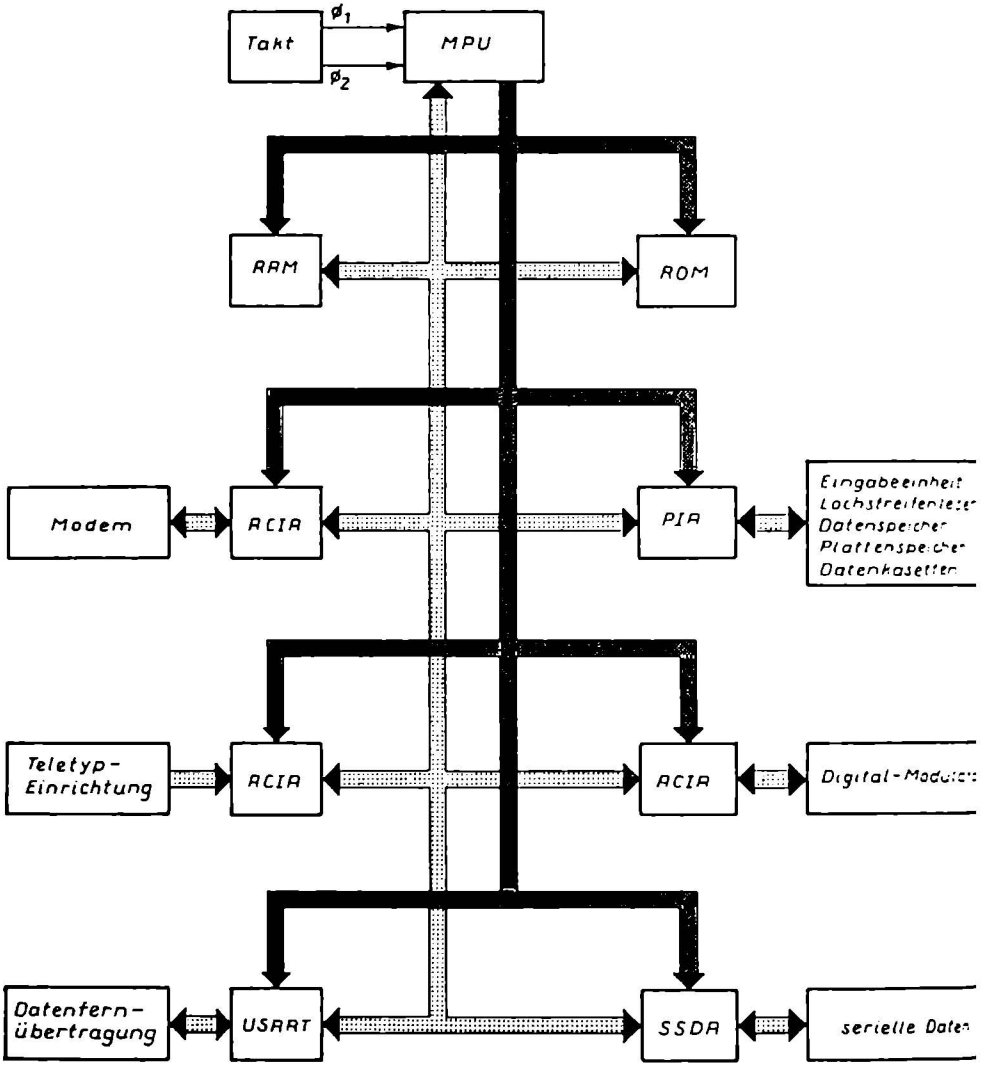


Bild 5.1 Aufbau eines Mikroprozessor-Systems

Takt: Die meisten Mikroprozessor-Systeme arbeiten heute mit einem Zwei-Phasen-Takt. Der Takt wird von einem Quarz-Generator erzeugt und die Taktfrequenz liegt in der Größenordnung zwischen 0,1 MHz und 10 MHz.

RAM: Schreib-Lese-Speicher (random access memory). Dieser Speicher nimmt die Zwischeninformationen auf und speichert sie solange, bis sie abgerufen werden. Die einzelnen Adressen erhält der Speicher über den Adressen-Bus. Nach dem Ansteuern erfolgt das Einschreiben der Information, die über den Daten-Bus anliegen. Das Auslesen von gespeicherten Informationen erfolgt auf dem gleichen Prinzip. Hier wird der Daten-Bus zum Auslesen verwendet.
Die Speicherkapazität von RAMs hängt von der Größe des Systems ab. Heute verwendet man Speicherkapazitäten von 4 x 256 Bits bis zu 16 x 65 536 Bits. Außerdem unterscheiden wir zwischen statischen und dynamischen RAMs.

ROM: Festwertspeicher (read only memory). Dieser Speicher enthält feste und nicht löschbare Informationen vom Anwender (Entwickler). In einem Festwertspeicher werden die einzelnen Unterprogramme festgelegt. Bei entsprechender Ansteuerung über den Adressen-Bus können über den Daten-Bus die einzelnen Programme freigegeben werden. Durch Auswechseln der ROMs können wir einfach ein Programm oder Unterprogramm ändern.
Die Speicherkapazität von ROMs hängt von der Größe des Systems ab. Heute verwendet man Speicherkapazitäten von 8 x 32 Bits bis zu 16 x 65 536 Bits. Außerdem unterscheiden wir zwischen den ROMs (maskenprogrammierbaren), den PROMs (programmierbaren) und den EPROMs (löschbaren). Die maskenprogrammierbaren ROMs haben den Vorteil, daß sie preisgünstig hergestellt werden können. Der Nachteil liegt aber in der umständlichen Maskenherstellung für jedes Programm. Für jedes Programm muß eine eigene Maske hergestellt werden.
Bei den programmierbaren Festwertspeichern kann sich der Anwender durch ein Programmiergerät sofort ein ROM herstellen. Das Programmieren eines PROMs dauert in der Praxis selten länger als eine Sekunde. Man verwendet heute zwei Programmierverfahren, das Widerstands-Verfahren und das Dioden-Verfahren. Beide Verfahren haben ihre Vor- und Nachteile.
Die löschbaren PROMs können nach dem Erstellen eines Programmes aus dem System herausgenommen und der Speicherinhalt gelöscht werden. Sie können anschließend durch Stromimpulse erneut programmiert werden. Die Löschung des Speicherinhaltes geschieht mittels UV-Licht und dauert in der Regel bis zu 30 Minuten.

PIA: Ein- und Ausgabe-Adapter (peripheral interface adapter). Dieser Adapter stellt eine Schnittstelle für das System dar. Meistens wird hier ein programmierbarer Baustein eingesetzt. Die Ansteuerung erfolgt über den Adressen-Bus. Je nach der Programmierung können damit periphere Geräte wie Eingabe-Einheiten (Lochstreifenleser, Datengeber, Datenspeicher oder Plattenspeicher) und Ausgabe-Einheiten (Datensichtgeräte, Datenspeicher, Tonbänder oder Plattenspeicher) angeschlossen werden.

ACIA: Adapter für spezielle Betriebsfunktionen (asynchronous communications interface adapter). Hier können zusätzliche Eingangsdaten in das System eingeschrieben werden. Hier werden die seriellen Eingangsinformationen in parallele Ausgangsinformationen für den Daten-Bus umgeformt. Die parallelen Informationen des Daten-Busses können durch den ACIA in serielle Ausgangsinformationen umgeformt werden. An dem ACIA kann ein Fernschreiber, A/D-Wandler oder andere Umformer angeschlossen werden.

MODEM: Modulator und Demodulator für die Übermittlung von Daten in der Nachrichtentechnik. Die parallelen Daten des Mikroprozessor-Systems können in analoge Ausgangsspannungen umgewandelt werden. Die parallelen Daten werden dabei auf eine bestimmte Trägerfrequenz aufmoduliert. Dadurch ergeben sich bessere Übertragungseigenschaften. Durch den Demodulator können die eingehenden Daten von der Trägerfrequenz entkoppelt werden und die Daten liegen nach dem MODEM parallel in dem Mikroprozessor-System.

USART: Sender und Empfänger für die serielle Datenübertragung (universal synchronous/asynchronous receiver/transmitter) ohne Trägerfrequenz. Der Baustein arbeitet als Sender und Empfänger. Als Sender verarbeitet er die Informationen des Daten-Busses zu seriellen Daten um. Gleichzeitig ist er in der Lage als Empfänger zu arbeiten. Serielle Daten werden empfangen und für den Daten-Bus in parallele Daten umgeformt.

SSDA: Synchroner Serien-Daten-Adapter (synchronous serial date adapter). Statt dem ACIA kann auch dieser Baustein in einem System Verwendung finden. Über den Daten-Bus-Ein/Ausgang lassen sich die Daten in das System einschreiben oder auslesen. Die Eingangsinformationen der Peripherie laufen dabei über einen FIFO. Durch diese Einrichtung ist ein einwandfreies Arbeiten des Systems gewährleistet. Parallel zu diesem FIFO liegt ein Ausgabe-FIFO. Hier werden die Daten-Bus-Informationen für die angeschlossene Peripherie synchronisiert und zur Weiterverarbeitung freigegeben.

6. Mikroprozessor-System 6800

In diesem Kapitel sollen nur kurz die einzelnen Einrichtungen für das Mikroprozessor-System 6800 beschrieben werden.

6.1. MPU 6800

In Bild 6.1 ist die Innenschaltung des Mikroprozessors 6800 der Firma Motorola gezeigt. Der 6800 ist ein 8-Bit-Mikroprozessor und besitzt einen Befehlsvorrat von 72 Befehlen, wenn wir die verschiedenen Adressierungsarten mit betrachten, ergibt sich ein Befehlsvorrat von 197 Befehlen. Das erste Byte gibt den Operationscode an. Mit diesem Byte können wir den entsprechenden Befehl, also eine Operation durchführen und die Adresse bestimmen.
Durch die 16 Adressen-Ausgänge können wir ein Speichersystem von 64 k Bytes (≙ 65 536 Speicherplätze) ansteuern. Bei der Einadreß-Maschine haben wir dagegen nur 16 Speicherplätze zur Verfügung gehabt.

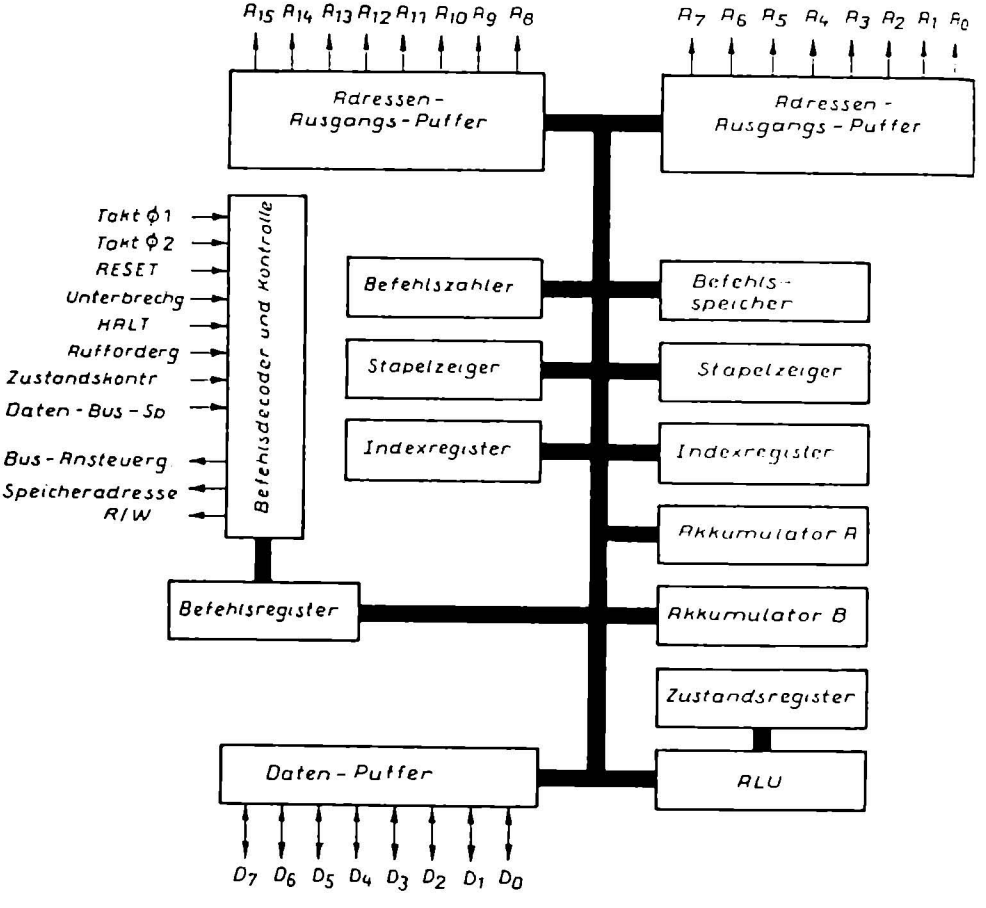


Bild 6.1 Innenschaltung des Mikroprozessors 6800

6.1.1 Register bei der MPU 6800

Der Mikroprozessor 6800 hat folgende Register:
a) Akkumulator A
b) Akkumulator B
c) Indexregister
d) Befehlszähler
e) Stackregister
f) Bedingungsregister

Die beiden Akkumulatoren in Bild 6.1 sind die Rechenregister in der 6800. Hier können wir bis zu 8-Bit verarbeiten. Die beiden Register sind gleichwertig und werden mit ACCA (Akkumulator A) und ACCB (Akkumulator B) bezeichnet.
Bei allen Rechenoperationen mit zwei Operanden soll der erste Operand in einem der beiden Akkumulatoren stehen. Der andere Operand kann in dem anderen Akkumulator stehen oder aus einer Adresse geholt werden:

I:	ACCA	→	L L H L L H H L	ACCA + ADD 14
	ADD 14	→	H L L L L H L L	
II.	ACCA	→	H L L L H H H L	ACCA + ACCB
	ACCB	→	L L L L H H L L	

Bei der ersten Möglichkeit haben wir eine normale Addition. Der erste Summand (erste Operand) steht im Akkumulator A. Durch einen Befehl holen wir aus der Adresse 14 den zweiten Summanden (zweite Operand) und addieren beide in der ALU.
Die zweite Möglichkeit wird meistens bei der Multiplikation verwendet. In dem Akkumulator A steht der Multiplikand. In dem Akkumulator B der Multiplikator. Über den Programmzähler können wir die Multiplikation durchführen.
In den beiden Akkumulatoren werden insbesondere die Addition und die Subtraktion durchgeführt. Die beiden können aber auch für logische Verknüpfungen verwendet werden.

6.1.2 Indexregister

Das Indexregister hat eine Wortlänge von 16 Bits. Damit können wir die Adressen zwischen 0 und 65 535 voll erfassen. Außerdem haben wir die Möglichkeit, die 16-Bits bei bestimmten artithmetischen Rechenoperationen einzusetzen.

6.1.3 Befehlszähler

Der Befehlszähler umfaßt ebenfalls eine Wortlänge von 16-Bits. Die Befehle, aus denen unser Programm zusammengesetzt ist, sind im Befehlsspeicher des 6800 enthalten. Der Mikroprozessor adressiert den Speicher um festzustellen, welche Funktion als nächster Programmschritt auszuführen ist. Er muß also jeweils wissen, unter welcher Adresse der nächste Befehl zu finden ist.
Jeder Speicherplatz ist numeriert, um ihn von allen anderen Speicherplätzen zu unterscheiden. Die Zahl, welche einen Speicherplatz identifiziert, wird seine Adresse genannt. Der Mikroprozessor ist mit einem Befehlszähler ausgestattet, der jeweils die Adresse für den nächsten Programmbefehl enthält. Jedesmal, wenn der 6800 sich einen Befehl einholt, läßt er seinen Befehlszähler um einen Schritt weiterzählen, so daß dieser immer auf den nächsten Befehl ausgerichtet ist.

6.1.4 Stackregister

Das Stackregister oder der Stapelzeiger ist für die Subroutine erforderlich. Eine Subroutine ist ein Unterprogramm, d.h. wenn wir in unserem Programm ein Unterprogramm eingeben, so verläßt der Mikropro-

zessor sein Hauptprogramm und wechselt in ein bestimmtes Unterprogramm über. Bei dem Überwechseln handelt es sich immer um einen Sprungbefehl. Bei einem Sprungbefehl muß der Mikroprozessor aber den Inhalt seines Befehlszählers zum Zeitpunkt seines Sprunges festhalten. Ist das Unterprogramm beendet, so kann der Mikroprozessor auf dem festgehaltenen Inhalt des Befehlszählers weiterarbeiten. Somit stellt ein Unterprogramm ein abgeschlossenes Programm dar.
Unterprogramme erzeugen wir, wenn sich das Hauptprogramm wiederholen muß, wie die Berechnung für einen Sinus oder den Logarithmus einer Variablen.
In den Stapelzeiger können wir neben Unterprogrammen auch einen Interrupt oder eine Einzelbefehlsausführung durchführen.

6.1.5 Befehlsliste für den 6800

ABA Addiere Akkumulatoren
ADC Addiere mit Übertrag
ADD Addiere
AND Logische UND-Verknüpfung
ASL Arithmetisches Links-Schieben
ASR Arithmetisches Rechts-Schieben
BCC Sprung wenn kein Übertrag
BCS Sprung wenn Übertrag
BEQ Sprung wenn Null
BGE Sprung wenn größer oder gleich Null
BGT Sprung wenn größer Null
BHI Sprung wenn größer
BIT Bit-Test
BLE Sprung wenn kleiner oder gleich
BLS Sprung wenn kleiner oder gleich
BLT Sprung wenn kleiner Null
BMI Sprung wenn nicht Null
BPL Sprung wenn Plus
BRA Unbedingter Sprung
BSR Sprung in das Unterprogramm
BVC Sprung wenn kein Überlauf
BVS Sprung wenn Überlauf
CBA Vergleiche Akkumulatoren
CLC Lösche Übertrag
CLI Lösche Unterbrechungs-Maske (Interrupt)
CLR Lösche
CLV Lösche Überlauf
CMP Vergleiche
COM Negation für logische Funktionen
CPX Vergleiche Indexregister
DAA Dezimalangleichung
DEC Vermindere um −1
DES Vermindere Stapelzeiger um −1
DEX Vermindere Indexregister um −1
EOR Exklusiv ODER
INC Erhöhe um +1
INS Erhöhe Stapelzeiger um +1
INX Erhöhe Indexregister um +1
JMP Sprung
JSR Sprung in das Unterprogramm
LDA Lade in Akkumulator
LDS Lade in Stapelzeiger
LDX Lade in Indexregister
LSR Daten nach rechts schieben

NEG Negation für arithmetische Funktionen
NOP Keine Operationen
ORA Inklusiv ODER
PSH Transport vom Akkumulator in den Stapelzeiger
PUL Transport vom Stapelzeiger in den Akkumulator
ROL Daten nach links umlaufen
ROR Daten nach rechts umlaufen
RTI Rücksprung aus einer Unterbrechung
RTS Rücksprung von einem Unterprogramm
SBA Subtraktion der Akkumulatoren
SBC Subtraktion mit einem Übertrag
SEC Setze Übertrag
SEI Setze Unterbrechungs-Maske
SEV Setze Überlauf
STA Speichere Akkumulator
STS Speichere Stapelzeiger
STX Speichere Indexregister
SUB Subtrahiere
SWI Unterbrechung durch die Software
TAB Transport Akkumulatoren
TAP Transport Akkumulator nach Zustandsregister
TBA Transport Akkumulatoren
TPA Transport vom Zustandsregister nach Akkumulatoren
TST Test
TSX Transport vom Stapelzeiger nach Indexregister
TXS Transport vom Indexregister in den Stapelzeiger
WAI Warten auf Unterbrechung (Interrupt)

6.2. Taktgeber für den Mikroprozessor 6800

In Bild 6.2 ist eine einfache Schaltung für einen Taktgeber gezeigt. Zwei Monoflops sind zu einem einfachen Taktgenerator zusammengeschaltet. Die Taktfrequenz dieses Generators liegt bei 200 kHz. Das linke Monoflop steuert mit seinem $\overline{Q}$-Ausgang eine Transistor-Treiberschaltung an. Diese Treiberstufe erzeugt den Takt $Ø_1$. Das rechte Monoflop steuert die untere Transistor-Treiberstufe an und damit haben wir den anderen Takt $Ø_2$. In Bild 6.2 ist noch ein weiterer Ausgang, der $Ø_2$-SYNC gezeigt. Dieser Ausgang ist mit allen E-Eingängen (CSO) des Mikroprozessor-Systems verbunden, außer mit der 6800.

6.3. Der Schreib-Lese-Speicher (RAM)

In diesem Buch soll nur kurz die Funktionsweise eines Schreib-Lese-Speichers beschrieben werden. Nähere Angaben finden Sie im Mikroprozessor-Buch Teil 2.
Ein RAM ist ein Speicherbaustein, in dem wir die Programmdaten, Übersichtstabellen, Zwischenwerte oder Stapeldaten abspeichern können. Wird die Spannungsversorgung bei einem RAM abgeschaltet, so geht der gesamte Speicherinhalt verloren. Wenn Programme oder Tabellen-Daten gespeichert sind, die unbedingt erhalten bleiben müssen, so werden diese Daten auf den Floppy-Disk gegeben. Ein Floppy-Disk ist ein schnelles Speichermedium, daß extern vom Mikroprozessor geschaltet ist und einen schnellen Zugriff erlaubt.

6.4. Der Programm-Speicher (ROM)

In dem Buch 2 über Mikroprozessoren finden Sie ausführlichere Daten über einen Festwertspeicher. Ein ROM ist ein Speicherbaustein, der die Programme und die Konstanten speichert. Der Inhalt eines ROMs ist nicht wie beim RAM flüchtig, wenn wir die Betriebsspannung abschalten. Der Inhalt eines ROMs bleibt immer erhalten. In Bild 6.4 ist ein ROM gezeigt.

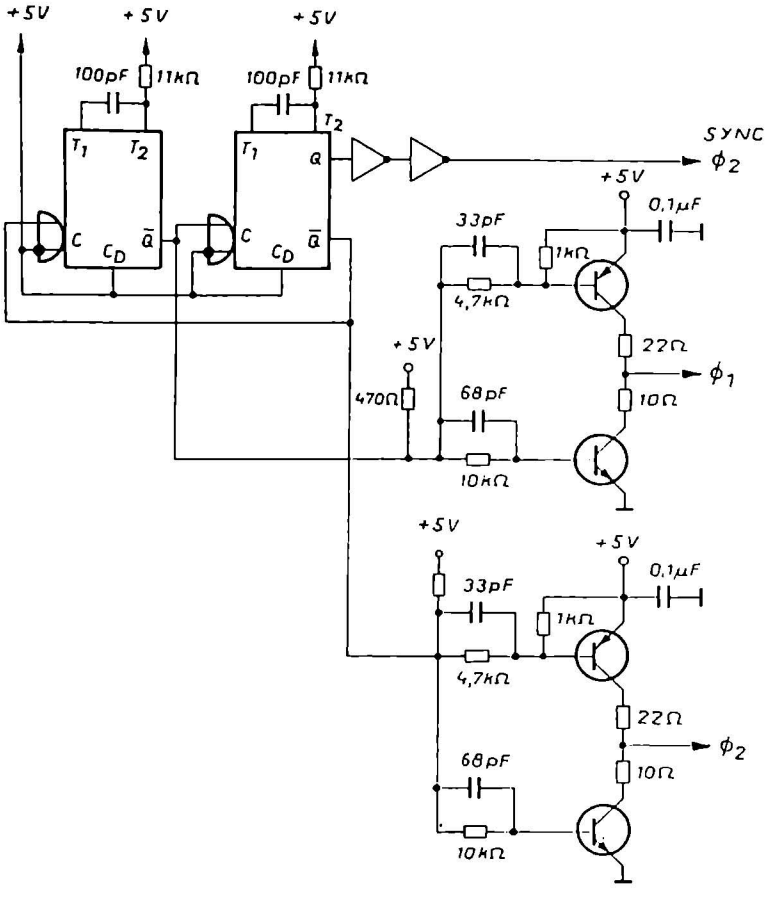


Bild 6.2 Taktgenerator für das Mikroprozessor-System 6800

6.5. Kurzbeschreibung eines peripheren Interface-Adapters (PIA)

Eine PIA stellt eine Schnittstelle zwischen dem Mikroprozessor und zwei Peripherie-Geräten dar. In Bild 6.5 ist der Aufbau gezeigt.

An dem Daten-Bus liegen die Eingangsdaten an. Gleichzeitig können über diese auch die Ausgangsdaten ausgelesen werden. Die Eingangsdaten können in das Eingangsregister gespeichert werden, wenn der R/W-Steuereingang ein L-Signal aufweist. Mit einem L-Signal erfolgt ein direkter Datentransport vom Mikroprozessor über den Daten-Bus in das Eingangsregister. Die Daten werden gespeichert und auf den PIA internen Eingangs-Bus gegeben.

Die Datenausgabe erfolgt durch die Ansteuerung der CA_1- und CB_1-Eingänge. Hat der CA_1-Eingang ein H-Signal, so werden die Daten über das Peripherie-Interface A ausgelesen. Bei einem H-Signal am CB_1-Eingang kann die Datenausgabe über das Peripherie-Interface B erfolgen.

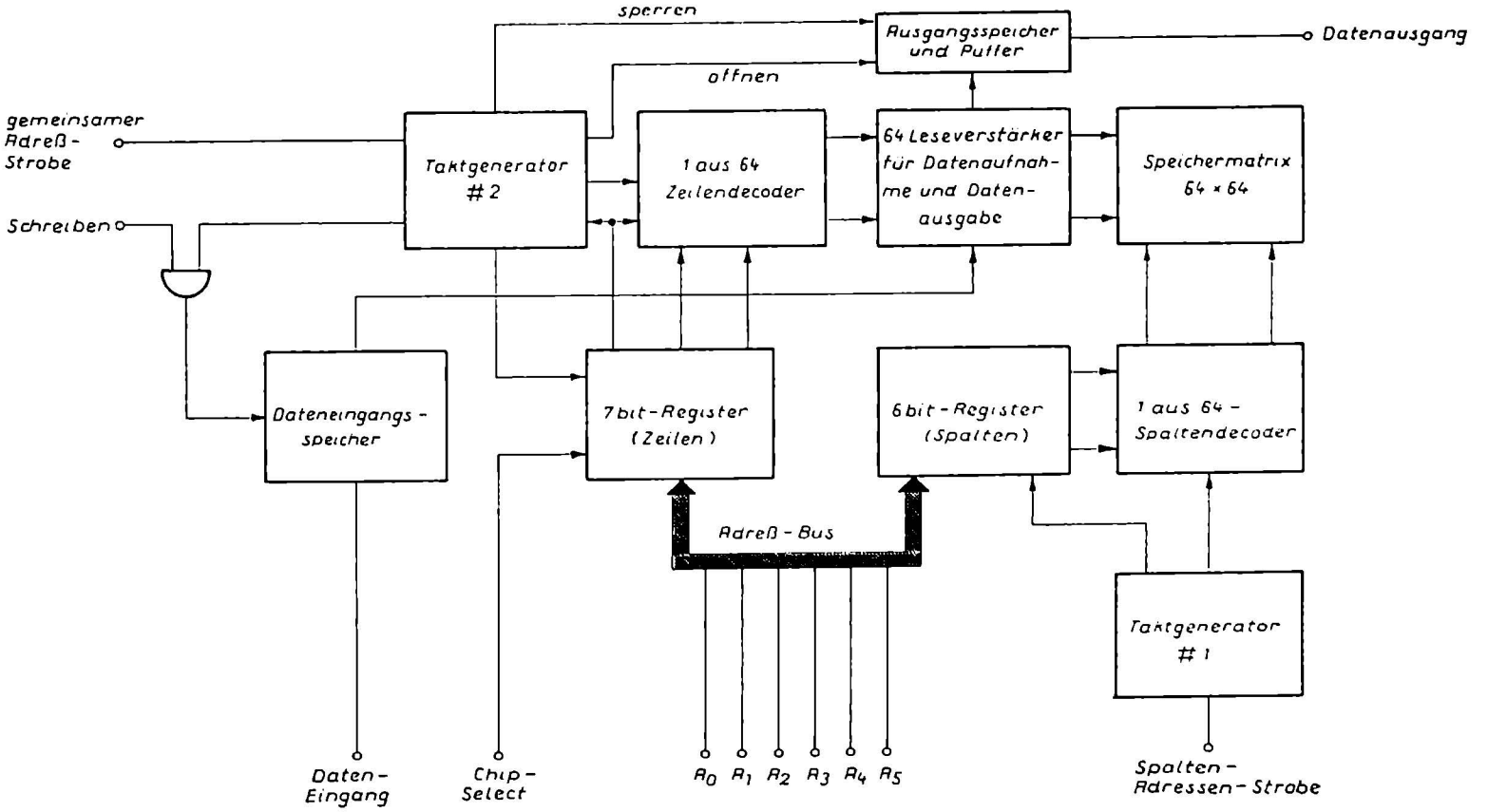


Bild 6.3 Aufbau eines dynamischen RAMs mit 4096-Bits

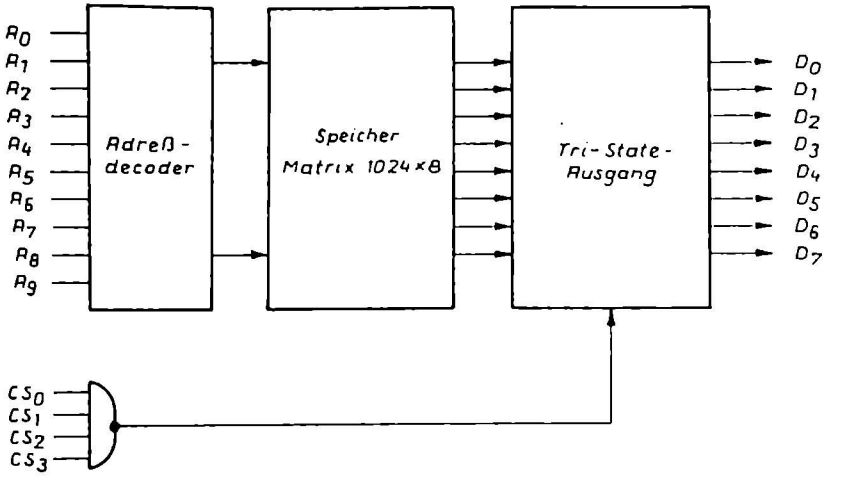


Bild 6.4 Aufbau eines ROMs mit 1024 x 8-Bits

Wesentlich dabei ist der Signalzustand an dem CA_2- und CB_2-Eingang. Mit diesen beiden Eingängen kann das Programm unterbrochen werden. Dies gilt sowohl für den Ausgabe-Betrieb, also wenn die Daten ausgelesen werden sollen, als auch für den Eingabe-Betrieb. Mit einem H-Signal an dem CA_2 können die Daten über die PA-Ausgänge ausgelesen und der Mikroprozessor-Betrieb dabei unterbrochen werden, da der IRQA-Ausgang ein Signal führt.
Durch die PIA wird jedem Peripherie-Gerät ein spezielles Programm zugeordnet, daß dessen spezifische Funktionen berücksichtigt. Durch die CA_1, CA_2, CB_1 und CB_2 und den beiden Ausgängen IRQA und IRQB kann eine Programm-Unterbrechung (interrupt) erzeugt werden. Dies erhöht die Wirksamkeit des Mikroprozessor-Systems beträchtlich. Wir können bei einer Dateneingabe durch einen Interrupt-Befehl den Mikroprozessor direkt ansteuern und die Daten eingeben. Wir können die Daten aus dem Mikroprozessor direkt in ein Peripherie-Gerät transportieren. Hierzu halten wir den Mikroprozessor kurz an.
Nehmen wir an, ein Mikroprozessor ist mit der Verarbeitung einer großen Datenmenge beschäftigt. Von dieser Datenmenge soll ein Teil in den Drucker gegeben werden. Der Mikroprozessor kann ein Datenwort innerhalb eines einzelnen Operationszyklus ausgeben. Der Drucker benötigt aber wesentlich länger, dieses Datenwort auszudrucken. Normalerweise geht der Mikroprozessor in eine Wartestellung, bis der Drucker das nächste Datenwort empfangen kann. Um dies zu vermeiden, gibt der Mikroprozessor sein Datenwort auf den Daten-Bus und dieser transportiert dieses zur PIA. In dem Eingangsregister der PIA wird dieses Datenwort gespeichert und über das Peripherie-Interface A oder B ausgegeben. Gleichzeitig erhält der Mikroprozessor über einen der PIA-Kontrollregister und der entsprechenden Unterbrechungs-Zustandskontrolle (interrupt status controll) einen Befehl und der Mikroprozessor kann sofort weiterarbeiten.
Hat der Drucker das Datenwort verarbeitet, gibt es an einen CA_1 oder CB_1-Eingang ein Signal ab. Die PIA sendet einen Interrupt aus, der Mikroprozessor quittiert diese Aufforderung, unterbricht die Bearbeitung des Hauptprogrammes und wechselt automatisch über in ein Unterprogramm, welches das nächste Datenwort gespeichert hat. Dieses Datenwort wird auf den Daten-Bus gegeben, von der PIA zwischengespeichert und an den Drucker weitergegeben. Nachdem die PIA das Datenwort gespeichert hat, gibt es über das Kontrollregister und der Unterbrechungs-Zustandskontrolle einen Weiterverarbeitungsbefehl ab. Der Mikroprozessor kehrt danach in sein Hauptprogramm zurück.

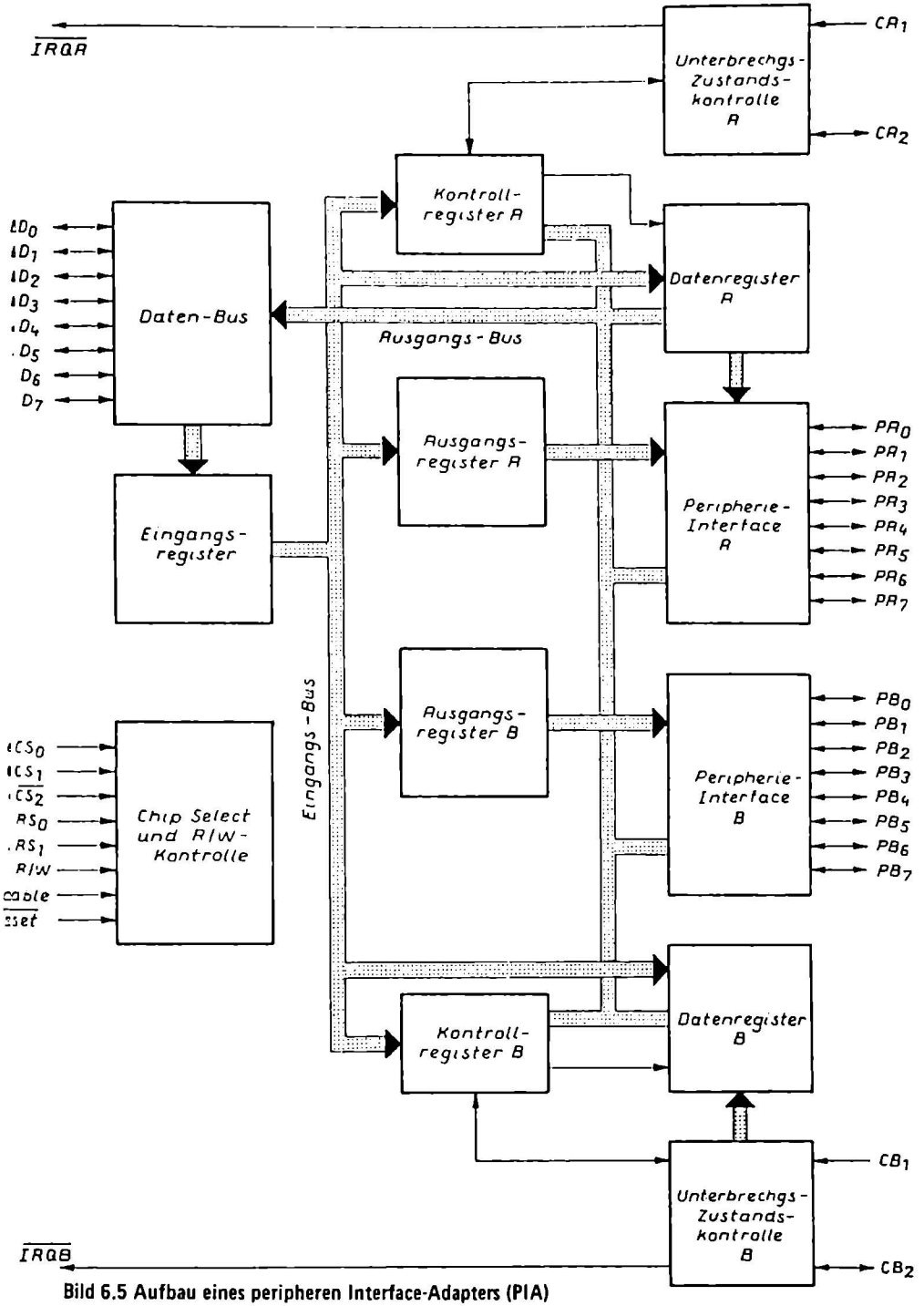


Bild 6.5 Aufbau eines peripheren Interface-Adapters (PIA)

6.6. Kurzbeschreibung eines asynchronen Interface-Adapters (ACIA)

In Bild 6.6 ist der Aufbau eines ACIA gezeigt. Der Baustein erfüllt mehrere Funktionen. Ein paralleles 8-Bit-Datenwort kann von der ACIA in ein serielles Datenwort umgesetzt werden. Die seriellen Daten können wir an dem Sendedaten-Ausgang abnehmen. Über den Empfängerdaten-Eingang ist das ACIA in der Lage, serielle Datenwörter in 8-Bit-Datenwörter mit paralleler Struktur umzuwandeln.
Mit dem R/W-Eingang können wir bestimmen, ob der ACIA senden oder empfangen soll. Bei einem H-Signal sendet der Baustein. Die Datenwörter am Eingang werden nacheinander in das Sendedaten-Register übernommen und gespeichert. Der Ausgang des Sendedaten-Registers ist mit dem Sendeschiebe-Register verbunden. Mit der Taktfrequenz vom Sendetakt werden die Informationen nacheinander eingeschoben. Das Aussenden der Daten geschieht über den Sendedaten-Ausgang.
Wenn wir einen Drucker ansteuern, können wir von dem Drucker aus die Daten des Sendeschiebe-Registers extern löschen. Sendet der ACIA, geht der Ausgang „Beginn zum Senden" auf Signal und über den Sendedaten-Ausgang verlassen die Daten den ACIA. Das Senden der Daten ist asynchron.
Wenn der ACIA Daten empfangen soll, geben wir diese Daten auf den Empfängerdaten-Eingang. Mittels dem Empfangstakt können wir nacheinander die seriellen Daten in das Empfangsschiebe-Register einspeichern. Die Daten werden anschließend im Empfangsdaten-Register gespeichert und auf den Daten-Bus gegeben.
Durch die Programmierung des Kontrollregisters über einen Befehl, können wir bestimmen, ob ein 1-, 16- oder 64-Taktbetrieb erfolgen soll. Dadurch ist der Baustein in der Lage, bis zu 500 000 Datenwörter in der Sekunde zu übertragen.

6.7. Kurzbeschreibung eines synchronen seriellen Daten-Adapters (SSDA)

In Bild 6.7 ist der Aufbau eines SSDA gezeigt. In der Funktionsweise ist der SSDA mit dem ACIA fast identisch. Die Betriebsart beim SSDA ist aber synchron.
Die Eingangsdaten beim Sender-Betrieb liegen an den Daten-Eingängen D_0 bis D_7 an. Diese Daten werden anschließend in einen FIFO gegeben. Ein FIFO ist ein Spezialspeicher. Die eingegebenen Daten werden durch den FIFO zwischengespeichert und nacheinander in das Übertrager- oder Sende-Schieberegister geschoben. Der Ausgang des Schieberegisters sendet dann die Daten.
Der Empfang der Daten geschieht in umgekehrter Reihenfolge. An dem Empfänger-Eingang liegen die Daten an. Die Daten werden in dem Empfänger-Schieberegister zwischengespeichert und auf den FIFO gegeben. Der FIFO schiebt anschließend die Daten parallel auf den Daten-Bus.
Die synchrone Arbeitsweise des SSDA kann nur durch die FIFOs erreicht werden. Der FIFO speichert seine Daten so, daß sie immer sofort am Ausgang anstehen. Geben wir auf einen leeren FIFO am Eingang eine Information, so wird diese Information sofort zum Ausgang transportiert. Es erfolgt also keine Zwischenspeicherung, wie es beim Schieberegister der Fall ist. Die anderen Daten, die in einem FIFO gespeichert werden sollen, werden nacheinander am Ausgang gespeichert. Durch den FIFO ergibt sich nicht nur eine synchrone Arbeitsweise des SSDA, sondern auch eine hohe Datenverarbeitungsgeschwindigkeit, da empfangene Datenwörter sofort auf den Ausgang geschaltet werden. Eine Zwischenspeicherung entfällt.
Mikroprozessoren haben eine hohe Rechengeschwindigkeit. Nehmen wir wieder folgendes Beispiel an Ein Mikroprozessor ist mit einer großen Datenmenge beschäftigt, von dem ein Teil an einen Drucker gegeben werden soll. Der Mikroprozessor benötigt zur Ausgabe seiner Daten nur einen Bruchteil von dem Ausdrucken der Information durch den Drucker. Der Mikroprozessor würde in diesem Fall so lange stillstehen, bis der Drucker fertig ist. Wenn wir im Mikroprozessor eine Möglichkeit zur Unterbrechung (interrupt) des Programmes vorsehen, kann der Mikroprozessor seine Datenwörter aussenden. Diese Datenwörter würden dann im FIFO gespeichert werden. Nach dem Aussenden der Datenwörter kann der Mikroprozessor auf sein Hauptprogramm zurückkehren und weiterarbeiten.

Im Gegensatz zum ACIA haben wir alle Datenwörter in dem SSDA gespeichert. Der Drucker holt sich aus dem FIFO seine Datenwörter nacheinander heraus. Ist der FIFO leer und kann damit neue Datenwörter empfangen, so gibt der SSDA eine Unterbrechungsanforderung an den Mikroprozessor. Der

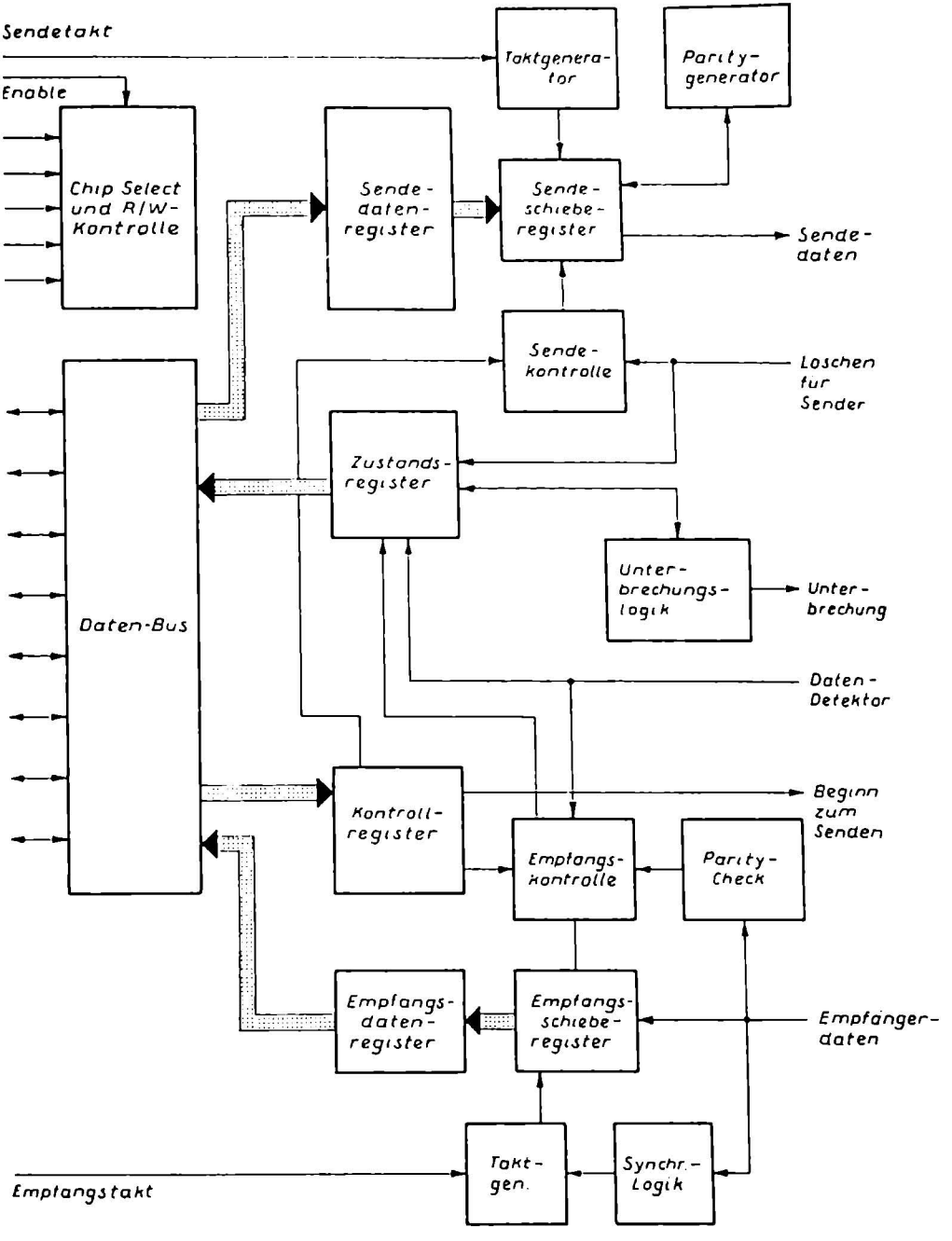


Bild 6.6 Asynchroner Interface-Adapter (ACIA)

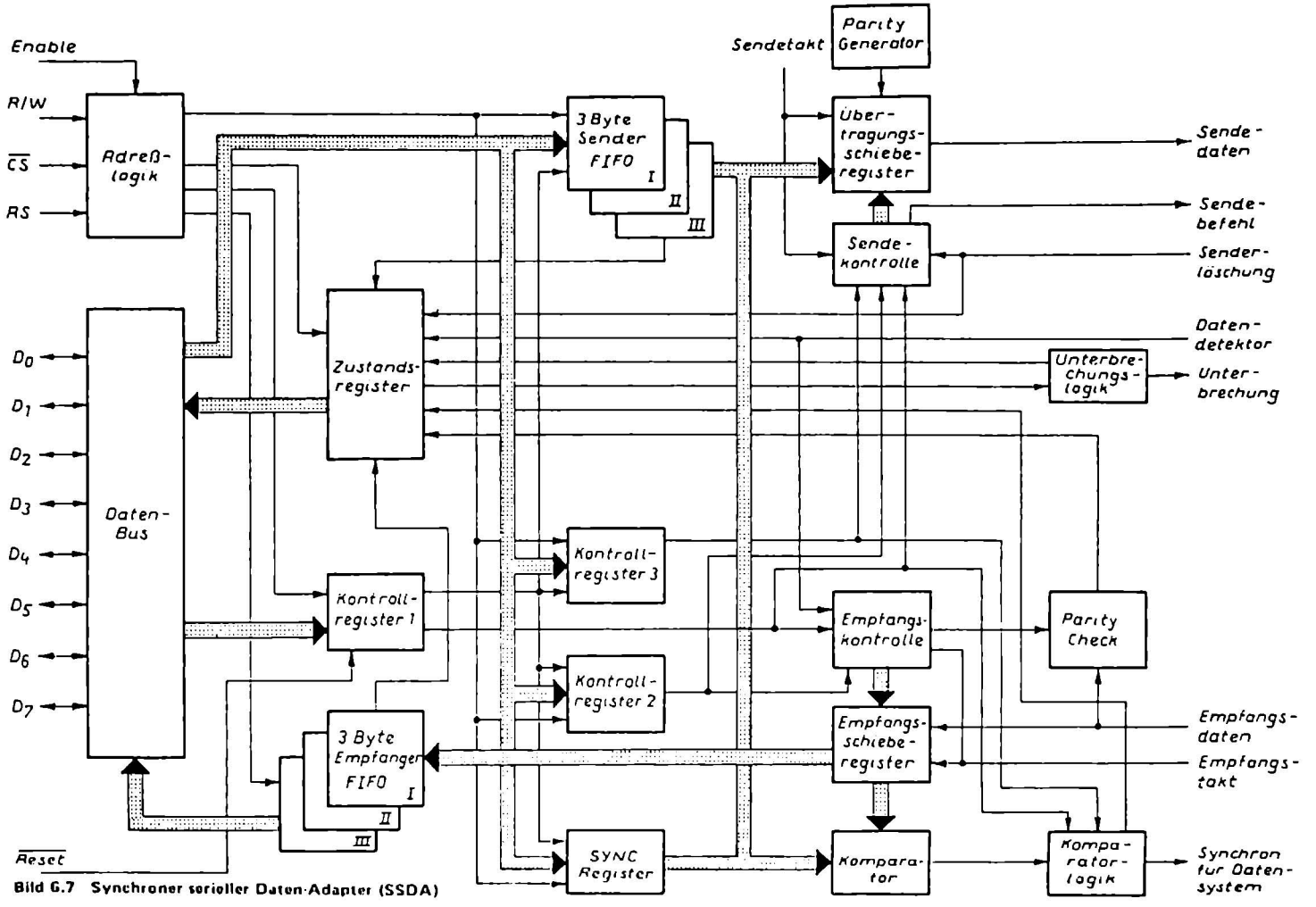


Bild 6.7 Synchroner serieller Daten-Adapter (SSDA)

Mikroprozessor quittiert diese Anforderung, unterbricht die Bearbeitung des Hauptprogrammes und wechselt automatisch in ein Unterprogramm, welches die anliegenden Datenwörter an den FIFO ausgibt. Eine Programmunterbrechung ist aber nicht bei der SSDA unbedingt erforderlich. Hat der Mikroprozessor ein Datenwort für den Drucker fertig, dann gibt er es gleich an den FIFO weiter. Der FIFO transportiert das Datenwort unmittelbar an das letzte Datenwort heran. Dies hat den Vorteil, daß zwischen den einzelnen Datenwörtern kein Zwischenraum vorhanden ist.
Die Betriebsweise beim Senden ist bei einem SSDA-Baustein sehr praktisch, da sich der Drucker aus dem FIFO direkt seine Datenwörter herausholen kann. Er zwingt damit den Mikroprozessor nicht in ein Unterprogramm zu gehen.
Das Empfangen von Datenwörtern geht ebenso schnell. Ist der FIFO für den Empfang leer und erhält der Eingang ein Datenwort, so wird das Datenwort sofort zum Ausgang geschoben. Die nachfolgenden Datenwörter reihen sich nacheinander in dem FIFO auf. Der Mikroprozessor ist nun in der Lage, nacheinander seine Datenwörter aus dem FIFO zu holen. Der Mikroprozessor benötigt also keine unnützen Wartezeiten, bis ein Peripherie-Gerät ein Datenwort übermittelt hat.

6.8. Kurzbeschreibung eines Modulators und Demodulators (MODEM)

In Bild 6.8 wird die Schaltung eines MODEM gezeigt. Ein MODEM ist immer zwischen einem ACIA-Baustein und einem elektronischen Meß- oder einer anderen elektronischen Einrichtung eingeschaltet. In der Regel wird der MODEM für die Umwandlung von Analogsignale in Digitalsignale benötigt. An dem I/O liegt eine Analogspannung an. Diese Spannung kann von einer Meßeinrichtung sein. Die Spannung liegt über den Koppler am Empfangsfilter. Das Empfangsfilter ist für einen Störabstand erforderlich. Über den Begrenzer wird das Analogsignal auf den Demodulator gegeben.
Analogspannungen können durch die Modulation auf eine Trägerfrequenz besser und damit genauer übertragen werden. Auf eine Trägerfrequenz wird die Meßspannung aufmoduliert und anschließend übertragen.

Der Demodulator hat die Aufgabe, die Meßspannung von der Trägerfrequenz zu trennen. Die Meßspannung liegt anschließend am Empfänger-Eingang des ACIA an. Durch den Schwellwert-Detektor kann die Meßspannung in eine digitale Informationsspannung umgewandelt werden. Hierzu bedient man sich der Impulsbreiten-Modulation. Je geringer die Meßspannung ist, desto kürzer die Ausgangsimpulse, d.h. das Tastverhältnis wird damit geändert. Ist die Meßspannung groß, so ist der Ausgangsimpuls wesentlich größer.
Setzt man die Impulsbreiten-Modulation in ein Zeitverhältnis, so kann der ACIA daraus bestimmte Datenwörter setzen, die dann der Mikroprozessor verarbeiten kann. Wesentlich ist hierbei der Takt- und Zeitablauf.
Die Daten des Mikroprozessors kommen über den ACIA-Baustein an das MODEM. Das MODEM moduliert diese Daten durch das Impulsbreiten-Modulations-Verfahren in eine Analogspannung um, die uns am I/O-Ausgang zur Verfügung steht.

6.9. Kurzbeschreibung eines universellen synchronen/asynchronen Empfänger/Sender-Bausteines (USART)

In Bild 6.9 ist die Schaltung eines USART gezeigt. Der USART wird vom Mikroprozessor wie jede andere Peripherie-Einrichtung behandelt und so programmiert, daß er praktisch mit jeder verwendeten Datenübertragung arbeiten kann.
Ein USART hat folgenden Vorteil. Der Mikroprozessor arbeitet in einer 8-Bit-Parallelstruktur. Der USART setzt diese Parallelstruktur in eine serielle Struktur um und sendet diese aus. Gleichzeitig kann er Daten in serieller Struktur empfangen und sie in eine parallele Struktur umwandeln. Der Mikroprozessor ist dann in der Lage, diese parallel anliegenden Daten zu verarbeiten.
Der USART-Baustein in Bild 6.9 hat zahlreiche Ein- und Ausgänge. Der Eingang „Datenübertragungs-Einrichtung" wird für die Bereitstellung der Sendeeinrichtung benötigt. Mit dem Eingang kann der Mikroprozessor durch einen Zustandslesebefehl die Sendeeinrichtung testen. Der nächste Eingang ist der Datenstations-Eingang. Mit diesem Eingang kann durch entsprechende Programmierung die Übertragungsgeschwindigkeit geändert werden. Durch ein Eingangssignal kann auch überprüft werden, ob die Datenstation senden kann.

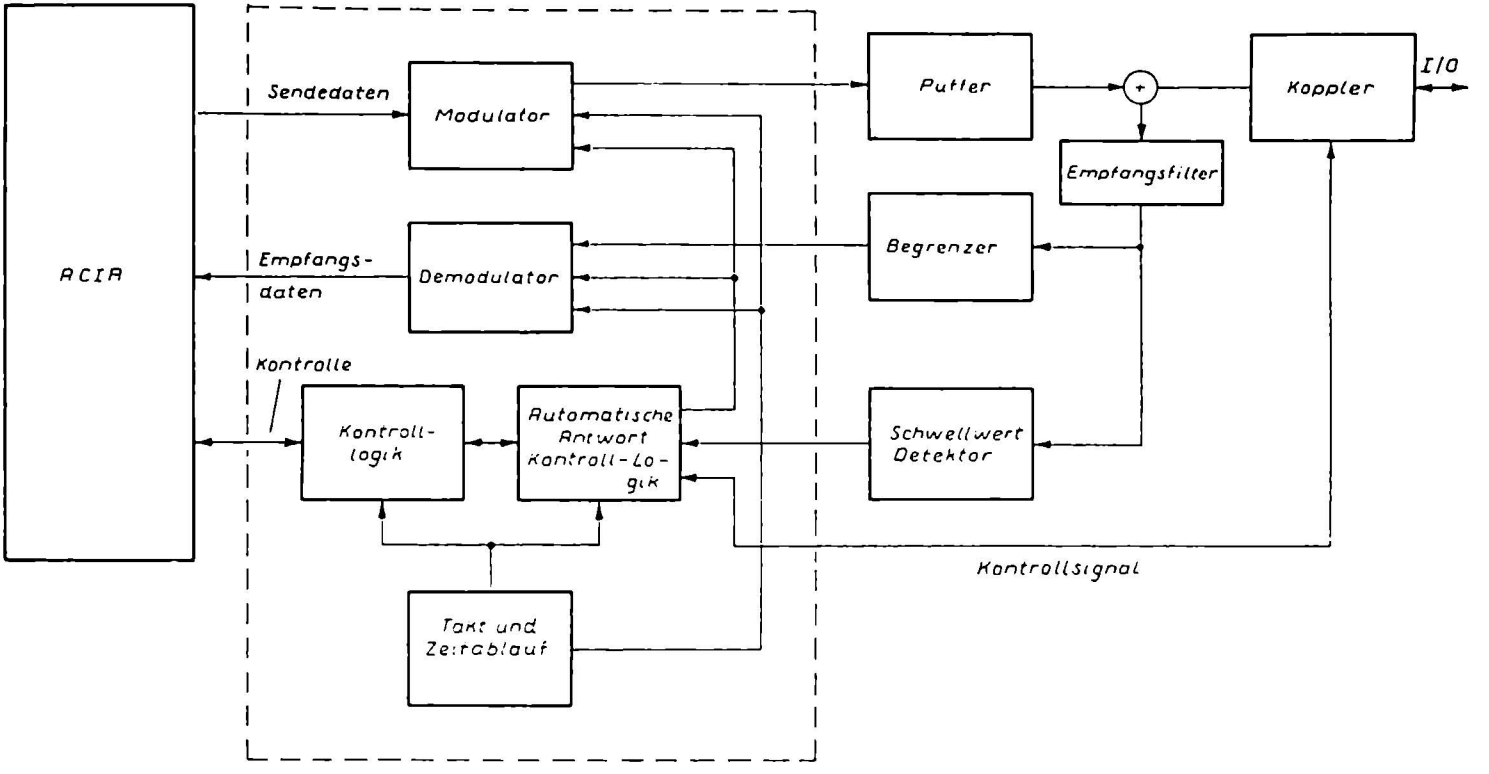


Bild 6.8 Aufbau eines MODEM

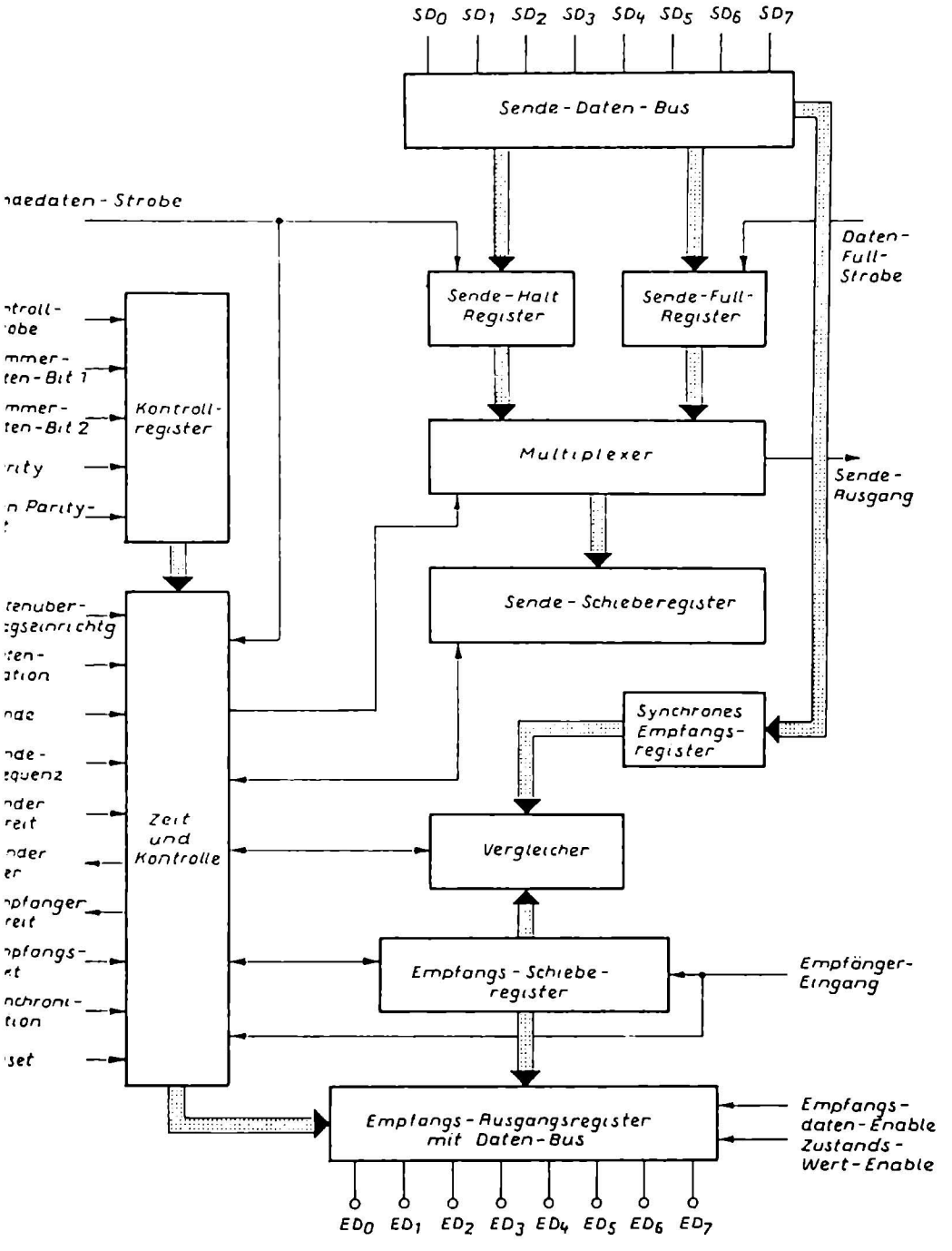


Bild 6.9 Aufbau eines Empfänger- und Sendebausteines

Mit dem Eingang „Senden" lösen wir im USART den Sendevorgang aus. Die Daten des Mikroprozessors liegen danach über den Sende-Daten-Bus und den entsprechenden Registerschaltungen am Ausgang an.
Mit dem Eingang „Sendetakt" bestimmen wir die Sendefrequenz.
Mit dem Ausgang „Sender bereit" kann der USART dem Mikroprozessor melden, daß der Sender zur Annahme eines Datenwortes bereit ist. Der gleiche Ausgang kann noch für die Unterbrechung des Systems herangezogen werden oder der Mikroprozessor prüft während der Aufforderung zum Laden mit einem Zustandslesebefehl. Der Ausgang „Sender bereit" wird nach dem Laden zurückgesetzt.
Sind im Sender keine Daten mehr vorhanden, so nimmt der Ausgang „Sender leer" ein H-Signal an Der Mikroprozessor erhält dadurch den Befehl ein neues Datenwort an den Sende-Daten-Bus zu bringen. Steht das Datenwort an, so verändert der Ausgang „Sender leer" seinen Signalzustand.
Mit der Sendefrequenz bestimmen wir die Übertragungsgeschwindigkeit, mit dem die Zeichen ausgesendet werden. Bei einem synchronen Betrieb ist die Frequenz gleich der tatsächlichen Übertragungsgeschwindigkeit. Bei einem asynchronen Betrieb ist die Frequenz aber ein vielfaches der tatsächlichen Übertragungsgeschwindigkeit. Ein Teil des Befehls für die Betriebsart wählt den Wert des Faktors aus. Der Faktor der Übertragungsgeschwindigkeit kann das 1-, 16-, oder das 64-fache betragen. Die Übertragungsgeschwindigkeit von 1000 Bit/s beträgt bei einer Sendefrequenz von 1 kHz das 1 fache, bei 16 kHz das 16 fache und bei 64 kHz das 64 fache.
Der Baustein eignet sich für folgende Übertragungsgeschwindigkeiten:

0 bis 56 kBit/s beim synchronen Betrieb
0 bis 9,6 kBit/s beim asynchronen Betrieb

Über dem Empfänger-Eingang liegen die Daten an dem Eingangs-Schieberegister an. In dem Schieberegister werden die ankommenden seriellen Daten in parallele Datenwörter umgeformt. Für den richtigen Empfang und der richtigen Umsetzung sind mehrere Steuer-Eingänge erforderlich.
Über die Empfänger-Synchronisation wird die Empfangs-Einrichtung synchronisiert. Der Eingang „Empfänger bereit" weist darauf hin, daß der USART ein Datenwort enthält, welches nun an den Mikroprozessor gegeben werden kann. Der Eingang „Zustands-Wert-Enable" liefert die Möglichkeit zur Programmunterbrechung des Mikroprozessors. Eine andere Möglichkeit ist, daß ein Aufforderungsbetrieb den Zustand dieses Einganges mit einem Zustandslese-Befehl überprüft. Liest der Mikroprozessor das Datenwort aus, so ändert sich der Zustand des Einganges.
Der Empfangstakt bestimmt die Geschwindigkeit, mit der der Empfänger arbeiten kann. Bei einem synchronen Arbeitsbetrieb ist die Frequenz von dem Eingang gleich der tatsächlichen Übertragungsgeschwindigkeit. Bei einem asynchronen Betrieb liegt die Frequenz von dem Eingang ein Vielfaches über der tatsächlichen Übertragungsgeschwindigkeit.
Zu den meisten Datenübertragungssystemen führt der USART sowohl die Sende- als auch die Empfangsvorgänge auf einer einzigen Datenübertragungsleitung durch. Aus diesem Grunde sind Sende- und Empfangsgeschwindigkeit identisch. Die beiden Eingänge, Sendefrequenz und Empfangstakt, erfordern in diesem Fall die gleiche Frequenz oder den gleichen Takt.
Der Anschluß „Synchronisation" wird nur im synchronen Betrieb verwendet, sowohl als Eingang, als auch als Ausgang. Mit einem L-Signal wird der gesamte Zeit- und Kontroll-Block in der USART zurückgesetzt. Dadurch ist die Synchronisation in ihrer Ausgangslage. Wenn der Anschluß als Ausgang benutzt wird (interne Synchronisation), nimmt der Anschluß ein H-Signal an. Für den Empfangsbetrieb muß der USART ein Synchronisations-Signal erkannt haben. Ist er für zweifache Synchronisationszeichen programmiert, nimmt dieser Eingang ein H-Signal in der Mitte des letzten Bits des zweiten Synchronisationszeichen an. Bei Synchronisieren wird der Zustands-Wort-Enable automatisch zurückgesetzt.
Verwenden wir den Anschluß als Eingang (externe Synchronisation), beginnt der USART nach L-H-Flanke des Signals mit dem Zusammensetzen eines Datenzeichens, wenn die nächste H-L-Flanke von dem Empfangstakt auftritt. Sobald die Synchronisation erfolgt ist, kann das H-Signal des Eingangssignales entfallen. Die Dauer des H-Signales sollte mindestens eine Periode von dem Empfangstakt betragen.

7. Der Mikroprozessor 8080

7.1. Einführung

Der Typ 8080 ist ein Ein-Chip-Mikroprozessor mit einer 8-Bit-parallel-Struktur für allgemeine Anwendungsbeispiele. Die Daten und Zustandsinformationen werden auf einem 8-Bit-Daten-Bus (D_0 bis D_7) und die Adressen für die Speichereinheiten und Peripheriegeräte über einen separaten 16-Bit-Adressen-Bus (A_0 bis A_{15}) übertragen.

Der Typ 8080 hat sechs Zeitsignal- und Steuer-Ausgänge:

SYNC	=	Ausgang zum Synchronisieren von externen Einrichtungen
DBIN	=	Daten-Bus-Steuerung für externe Einrichtungen
WAIT	=	Ausgang für die Warte-Steuerung
WR	=	Schreib-Ausgang
HLDA	=	Ausgang für die Halt-Steuerung
INTE	=	Ausgang für die Unterbrechungs-Steuerung

Vier Steuer-Eingänge:

READY	=	Eingang für die interne Warte-Steuerung
HOLD	=	Eingang für die interne Halt-Steuerung
INT	=	Eingang für die interne Unterbrechungs-Steuerung
RESET	=	Zum Löschen und Zurückstellen des Mikroprozessors

Der Typ 8080 hat noch vier Betriebsspannungs-Eingänge (+12V, +5V, 0V und −5V) und zwei Takt-Eingänge ($Ø_1$ und $Ø_2$).

7.2. Aufbau des 8080

In Bild 7.2 ist das Blockschaltbild des 8080 gezeigt. Dieses Blockschaltbild zeigt vier große Funktionsblöcke:

7.2.1 Register-Matrix und Adressen-Logik
7.2.2 Arithmetik-Logik-Einheit (ALU)
7.2.3 Befehl-Register und Steuer-Logik
7.2.4 Daten-Bus-Puffer

7.2.1 Register-Matrix und Adressen-Logik

Dieser Block befindet sich auf der rechten Seite von Bild 7.2.1 und besteht aus 13 Teilen:
Befehlszähler (program counter, PC) mit 16 Bits,
Stapelzeiger (stack pointer, SP) mit 16 Bits,
Register-Teil mit acht Registern, B-, C-, D-, E-, H- und L-Register mit je 8 Bits und zwei Zwischenregister W- und Z-Register, mit je 8 Bits. Die Register sind paarweise gruppiert und sind für allgemeine Anwendungen.

In dem Befehls- oder Programm-Zähler (program counter) sind die Speicheradressen für die einzelnen Programmbefehle gespeichert. Ein Programm setzt sich in einem Mikroprozessor immer aus Befehlen zusammen, die im Befehls-Speicher oder Befehls-Register gespeichert sind. Der Befehls-Zähler adressiert nun den Befehls-Speicher, um so festzustellen, welche Operation durchzuführen ist.
Der Speicher-Zähler beginnt bei dem Zählerstand 0 und gibt einen Befehlsabruf an den Befehls-Speicher ab und zählt dann um 1 weiter. Nach einem Freigabe-Signal gibt der Speicher-Zähler einen neuen Be-

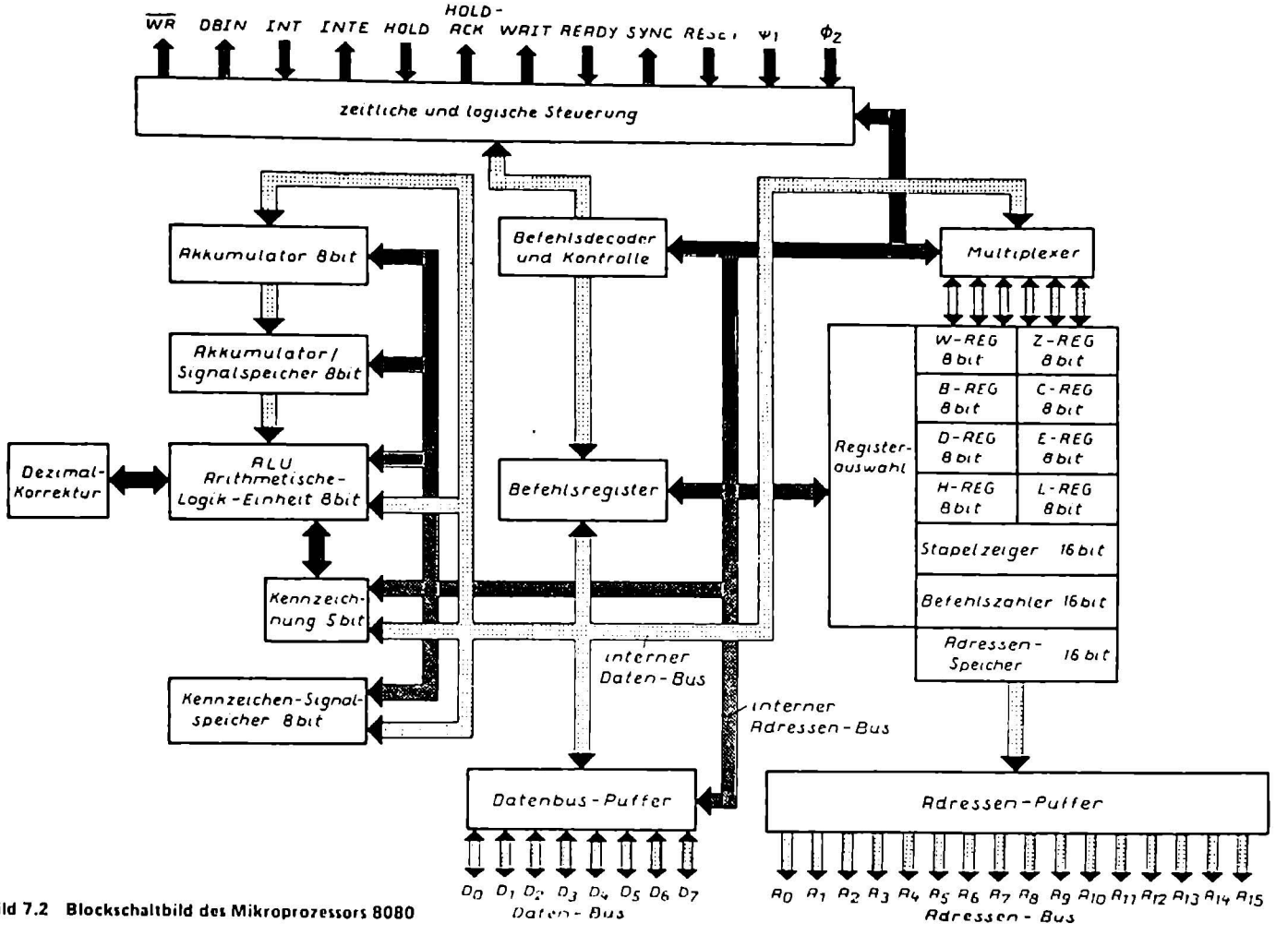


Bild 7.2 Blockschaltbild des Mikroprozessors 8080

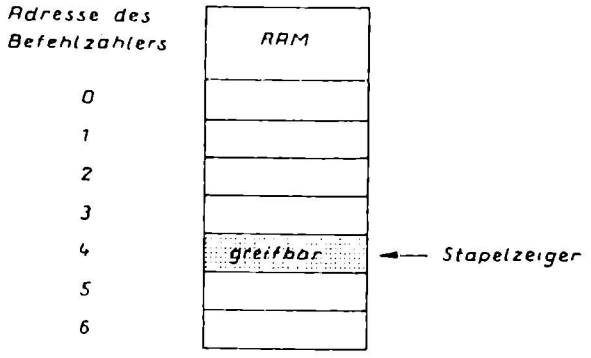


Bild 7.2.1 Prinzip des Stapelzeigers

fehlsabruf auf den Befehls-Speicher ab und zählt um 1 weiter. Wir haben damit den Zählerstand 2 im Befehls-Zähler. Da der Befehlzähler 16 Bits hat, ergibt sich ein Inhalt von 65 536 Befehlsabrufen für den 8080. Wir können in diesem Befehls-Zähler insgesamt 65 536 beliebige Befehle abrufen lassen.

Der Stapelzeiger (stack pointer) enthält die Adresse der nächsten Stapelstelle im Speicher. Der Stapelzeiger wird mittels dem Programm auf einen bestimmten Anfangswert gesetzt. Normalerweise beginnt man bei 0, aber wir haben die Möglichkeit durch die Bestimmung des Anfangswertes jeden Bereich des Schreib-Lese-Speichers als Stapel zu verwenden. In Bild 7.2.1 ist das Prinzip eines Stapelzeigers gezeigt. Durch die Adresse des Befehls-Zählers läuft das Hauptprogramm ab, das im RAM gespeichert ist. Bei der Adresse 4 wird das Hauptprogramm unterbrochen und eine Subroutine, ein Unterprogramm eingebaut. Das Unterprogramm ist ein Sprung-Befehl (jump) zu einem anderen Teil des Speichers. Ein Sprungbefehl enthält immer die Adresse des darauffolgenden Befehls.
Dieser folgende Befehl darf sich in einer beliebigen Speicheradresse befinden, sofern nur der programmierte Sprung die richtige Adresse angibt. Während des Ablaufes des Sprungbefehles ersetzt der Prozessor den Inhalt seines Programm-Zählers durch die Adresse, die in dem Sprungbefehl angegeben ist. Auf diese Art bleibt uns das Hauptprogramm erhalten.
In Bild 7.2.1 rufen (CALL) wir ein Unterprogramm ab. Bei diesem Sprung-Befehl hält der Mikroprozessor den Inhalt des Programm-Zählers fest. Dadurch können wir anschließend an das Unterprogramm sofort wieder an der richtigen Stelle unseres Hauptprogrammes beginnen. Das Unterprogramm stellt dadurch ein eigenständiges Programm innerhalb des Hauptprogrammes dar.
Das Unterprogramm besteht aus einer Vielzahl von Befehlen, die wir sonst im Hauptprogramm immer wiederholen müssen. Selbst bei kleineren Hauptprogrammen wird daher das Unterprogramm bereits eingesetzt. Funktionen, die oft im Unterprogramm geschrieben werden, sind die Berechnung eines Sinus oder Cosinus oder der Logarithmus einer Variablen. Unterprogramme werden vom Mikroprozessor so abgewickelt, daß ihm eine ordnungsgemäße Rückkehr in das Hauptprogramm möglich ist.
Wenn der Mikroprozessor einen Rufbefehl erhält, erhöht er die Zahl im Befehls- oder Programm-Zähler um 1 und speichert den Inhalt des Zählers in einem hierfür reservierten Speicherplatz mit der Bezeichnung Stapel-Speicher (stack) ab. Der Stapel-Speicher speichert damit die nächste Adresse ab, wenn das Unterprogramm beendet ist. Dadurch wird sofort nach Beendigung des Unterprogrammes im Hauptprogramm weiter gearbeitet. Der Mikroprozessor braucht seinen Einstieg in das Hauptprogramm nicht erst umständlich zu suchen. Der Stapel-Speicher hebt so die erste Adresse nach Beendigung des Unterprogrammes auf. Dadurch ergeben sich wesentlich höhere Geschwindigkeiten bei einem Programmablauf. Anschließend lädt der Mikroprozessor die Adresse, die in dem CALL-Befehl enthalten ist, in seinen Befehls- oder Programmzähler.

Der nächste eingeholte Befehl ist damit der erste Schritt in das Unterprogramm. Der letzte Befehl aus dem Unterprogramm ist der Rückkehr-Befehl (RETURN), der aber keine Adresse enthalten braucht. Erhält der Mikroprozessor seinen Rückkehr-Befehl aus dem Unterprogramm in das Hauptprogramm, so ersetzt er einfach den bestehenden Inhalt seines Programm-Zählers durch die im Stapelspeicher befindliche oberste Adresse. Durch die Setzung auf die oberste, also auf die letzte Adresse, wird der Mikro-

prozessor zur Wiederaufnahme seines Hauptprogrammes veranlaßt und zwar unmittelbar.
Wir können durch den CALL- und RETURN-Befehl Unterprogramme ineinander verschachteln. Mit einem CALL gehen wir in das erste Unterprogramm. Ist dieses beendet gehen wir mittels RETURN aus dem Unterprogramm, aber auf der nächsten Adresse im Hauptprogramm steht CALL für das nächste Unterprogramm. Eine Programmverschachtelung dieser Art ist möglich, solange der Mikroprozessor genügend Kapazität für die Speicherung der notwendigen RETURNs, sowie die erforderliche Logik besitzt. Die maximale Größe einer Verschachtelung ist von der Größe des Stapelspeichers abhängig. Hat ein Stapel-Speicher die Kapazität für drei Rückkehr-Befehle, so kann die Verschachtelung von Unterprogrammen erfolgen. Der 8080 verwendet einen reservierten Teil des RAM-Speichers als Stapelspeicher und verwaltet intern lediglich ein sogenanntes Stapelzeiger-Register (stack pointer), das dann die jeweilige Adresse der letzten Stapelspeicher-Einschreibung enthält.
Die Verschachtelung von Unterprogrammen in einem externen Stapelspeicher ist normalerweise unbegrenzt.
Der Stapelzeiger wird jedesmal hochgezählt, wenn Daten in den Stapel eingegeben werden und hochgezählt, wenn Daten aus dem Stapel entnommen werden. Das Dateneingeben bezeichnen wir als PUSH und das Entnehmen als POP. Das bedeutet für den Stapel ein wachsen nach unten.

In Bild 7.2 befinden sich im Register noch sechs allgemeine Register. Diese Register können wir entweder als selbständige 8-Bit-Register oder auch als 16-Bit-Registerpaare verwenden. Die beiden Zwischenspeicher-Register W und Z sind nicht durch das Programm adressierbar und sie werden daher nur für die interne Ausführung von Befehlen benutzt. 8-Bit-Worte aus dem Befehls-Bus können über den Registerauswahl-Multiplexer zwischen dem internen Daten-Bus und der Register-Matrix übertragen werden. 16 Bit-Übertragungen sind zwischen der Register-Matrix und dem Adressen-Signal-Speicher oder dem Auf/Abwärts-Zähler möglich. Der Adressen-Signal-Speicher empfängt die Daten von einem der drei Registerpaare und steuert damit den 16-Adressen-Puffer (A_0 bis A_{15}), sowie auch den Auf/Abwärts-Zähler. Der Auf/Abwärts-Zähler erhält seine Daten vom Adressen-Signal-Speicher und überträgt sie zu der Register-Matrix. Die 16-Bit-Daten können entweder hinauf- oder herunter gezählt oder einfach zwischen den Registern übertragen werden.

7.2.2 Arithmetik-Logik-Einheit (ALU)

Die ALU bildet den Mittelpunkt des 8080 und enthält folgende Register:

- Akkumulator mit 8-Bit,
- Zwischenspeicher-Akkumulator mit 8-Bit (ACT)
- Zustandsregister mit 5-Bit (flag register) für Null, Übertrag, Parität und Hilfs-Übertrag
- Zwischenspeicher-Register mit 8-Bit (TMP)

Alle rechentechnischen und logischen Probleme werden in der ALU durchgeführt, wie arithmetische, logische und umlaufende Operationen. Die ALU erhält die Daten von dem vorgeschalteten Zwischenspeicher-Register (TMP), dem Zwischenspeicher-Akkumulator (ACT) und dem Übertrags-Flip-Flop. Das Ergebnis der Rechenoperation kann entweder zum internen Daten-Bus oder zum Akkumulator übertragen werden. Von der ALU können wir auch die Daten an das Kennzeichenregister mit 5-Bits übertragen.
Die ALU besteht aus einem Addierer, mit dessen Hilfe wir den Inhalt zweier Register addieren können. Durch die entsprechende Programmierung können wir auch Routinen (Unterprogramme) für das Subtrahieren, Multiplizieren und Dividieren durchführen. Die Routine führen wir ein, damit das Hauptprogramm nicht belastet wird.
Neben den arithmetischen Rechenoperationen können wir auch mit der ALU logische Funktionen, wie die Daten komplementieren, negieren oder um eine Stelle verschieben, ausführen.
In der ALU sind sogenannte Zustands-Bits, die Flags enthalten. Die Flags geben uns über die Ergebnisse der arithmetischen und logischen Operationen Aufschluß. Die Flags geben außerdem Informationen ab, wenn ein Übertrag, eine Null, ein Vorzeichen oder eine Parität auftritt.

Übertrag	=	carry
Null	=	zero
Vorzeichen	=	sign
Parität	=	parity

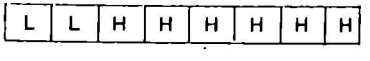
Wir können Programm-Sprünge mit den Flags durchführen. Hierzu führt das Programm einen bestimmten Sprung aus, wenn die Flags eine Kombination aufweist. Wir können das Programm in eine Routine (Unterprogramm) übergehen lassen, wenn z.B. ein Übertrag bei einer Addition auftritt. In der Routine würde z.B. stehen, addiere bei einem Übertrag die Summe mit einer dualen 6. Dadurch erhalten wir eine Summe im BCD-Code. Das Zwischenspeicher-Register (TMP) nimmt die Informationen des internen Daten-Busses auf und übergibt sie ganz oder nur teilweise an die ALU oder an das Zustands-Register. Der Akkumulator (ACC) kann von der ALU und vom internen Daten-Bus seine Informationen erhalten. Der Akkumulator überträgt seine Ergebnisse zum Zwischenspeicher-Akkumulator (ACT) und an den internen Daten-Bus. Der Inhalt des Akkumulators (ACC) und des Hilfs-Übertrags-Flip-Flops kann während der Durchführung eines DAA-Befehls (Akkumulators-Inhalt wird in eine zweistellige Zahl umgewandelt) auf die Dezimalkorrektur geprüft werden.

7.2.3 Befehls-Register und Steuer-Logik

Das Befehls-Register bezieht aus dem Daten- und Adressen-Bus seine Informationen. Normalerweise beginnt ein Programm mit der Adresse 0. Über den internen Daten-Bus kann das Befehls-Register seinen ersten Befehl aufnehmen, speichern und diesen an den nachfolgenden Befehlsdecoder und Operationszyklusdecoder abgeben.
Der Ausgang des Befehlsdecoders liefert uns mit verschiedenen Zeitimpulsen die Steuersignale für die Register-Matrix, die ALU und für die Daten-Puffer. Die Steuersignale steuern darüber hinaus noch mit den externen Steuersignalen zusammen den gesamten Steuerbereich der Schaltung an und liefern die Zustands- und Zyklus-Zeitsignale.
Über den internen Daten-Bus liegt ein 8-stelliges Wort an dem Befehlsregister an. Aus diesem 8-stelligen Wort kann der Mikroprozessor seine 78 Befehle entnehmen. Normalerweise können wir mit einem 8-stelligen Wort 256 Befehle übertragen, bei der 8080 werden aber nur 78 Befehle ausgeführt. Der Mikroprozessor liest in zwei Operationen einen Befehl (instruction-FETCH) ein. Zuerst sendet er die in seinem Befehlszähler befindliche Adresse zum Speicher. Dieser übermittelt daraufhin das adressierte By zum Mikroprozessor, der das empfangene Befehlswort in dem Befehls-Register speichert und es zur Steuerung der Abläufe während der nachfolgenden Befehlsausführung verwendet.

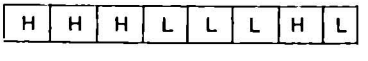
Die Umwandlung eines Befehlswortes in einen Programmablauf wird durch den Befehls-Decoder und den damit verbundenen Steuerschaltungen durchgeführt. Das Befehlswort wird von dem Befehls-Register gespeichert und von dem Befehls-Decoder decodiert. Damit kann dann der Mikroprozessor arbeiten.
In den meisten Anwendungen reicht ein 8-Bit-Befehlswort für eine Rechenoperation aus. In der Praxis kann es aber vorkommen, daß die Befehlsausführung mehrere Informationen benötigt, als wir in 8-Bits ausdrücken können. Dieser Fall tritt auf, wenn ein Befehlswort eine Operation und einen Speicherplatz beinhalten muß. Das einzelne Befehlswort kann die Operation durchführen, kann aber nicht die Adresse aufnehmen. Für diesen Fall müssen wir den Befehl aus 2 oder 3 Bytes zusammenstellen.

Ein-Byte-Befehl: Bei einem Ein-Byte-Befehl handelt es sich meistens um einen Operationsbefehl, wie z.B. arithmetische-, logische-, Speicherungs-, Routineanfang- oder Routineende-Befehle.
z.B.:

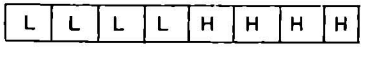


CMC-Befehl = Carry-Bit soll negiert werden

Zwei-Byte-Befehl: Bei einem Zwei-Byte-Befehl handelt es sich meistens um einen Operationsbefehl mit der Adresse für den Operanden.
z.B.:



JPO-Befehl = Bei Paritäts-Bit springe auf



ADR = Adresse 15

Wir haben hier einen Befehl für einen unmittelbaren Operationshinweis oder eine I/O-Operation (I = input, 0 = output, Ein/Ausgabe-Befehl)

Drei-Byte-Befehl: Die Adresse bei dem Zwei-Byte-Befehl reicht nur bis zur 255. Adresse. Mit der 8080 können 65 536 Adressen angesteuert werden. Hierzu ist daher ein Drei-Byte-Befehl notwendig. z.B.:

H	H	H	L	L	L	H	L
H	L	L	L	H	H	L	L
L	H	L	L	L	L	L	L

JPO-Befehl = Bei Paritäts-Bit springe auf

nADR = niederwertige Adresse 140

hADR = höherwertige Adresse 16 384

Die gesamte Adresse lautet: 16 384 + 140 = 16 524
Bei einem Paritäts-Bit springt der Mikroprozessor auf diese Adresse.

Aufeinanderfolgende Befehlsworte befinden sich im Speicher in hintereinanderliegenden Speicherplätzen und der Mikroprozessor erhält den vollen Befehl durch den 2 oder 3 FETCH-Zyklus. Unter einem FETCH verstehen wir einen Abruf. Das erste vom Speicher eingeholte Wort wird in dem Befehls-Register eingespeichert, das zweite Wort in einem Zwischenspeicher-Register, ebenso anschließend das dritte Wort. Nach der Abspeicherung der Worte beginnt der Mikroprozessor zu arbeiten.

Die Steuer-Logik des Mikroprozessors ist für den Ablauf eines Programmes erforderlich. Zusammen mit den Taktsignalen sorgt sie für den ordnungsgemäßen Ablauf des Programmes. Wird ein Befehl eingeholt und decodiert, so veranlaßt die Steuer-Logik durch das Aussenden eines Befehles an alle internen und externen Einheiten des Mikroprozessor-Systems die Aufnahme oder den Beginn einer Operation.
Die Steuer-Logik kann auch Befehle aufnehmen, z.B. eine Unterbrechung des Programmes oder eine Warte-Aufforderung. Bei einer Unterbrechungs-Aufforderung veranlaßt die Steuer-Logik eine Unterbrechung des Hauptprogrammes für die Bevorzugung eines Unterprogrammes. Nach der Beendigung eines Unterprogrammes gibt die Steuer-Logik das Hauptprogramm wieder frei. Eine Warte-Aufforderung (WAIT) wird oft von einem externen Speicher oder einer I/O-Einrichtung erteilt. Die Warte-Aufforderung kommt daher zustande, daß die externen Einrichtungen häufig langsamer als der Mikroprozessor arbeiten. Durch die Steuer-Logik bleibt der Mikroprozessor solange stehen, bis die externe Einrichtung die Daten aufgenommen oder abgegeben hat.

7.2.4 Verbindungen im Mikroprozessor

Die Verbindungen im Mikroprozessor sind der Daten-Bus, der Adressen-Bus und der Kontroll-Bus.
Über den Daten-Bus erhalten die einzelnen Einrichtungen im System ihre Daten. Da der 8080 einen Daten-Bus von 8-Bits hat, können wir ein achtstelliges Datenwort mit 256 Möglichkeiten für Befehle und Informationen transportieren.
Über den Adressen-Bus steuern wir die einzelnen Einrichtungen im System an. Bei der 8080 hat der Adressen-Bus eine 16-Bit parallele Struktur. Durch diese Struktur können insgesamt 65 536 Adressen angesteuert werden. Bei der Programmierung muß aber darauf geachtet werden, damit die Adressen nicht weit voneinander liegen. Jeder Programmsprung auf eine weit entfernte Adresse kostet Zeit!
Über den Kontroll-Bus erfolgt die Überwachungsfunktion der Steuer-Logik. Die Steuer-Logik ist über einen 6-Bit parallelen Bus mit den internen Einrichtungen verbunden.

Die Daten für den Mikroprozessor kommen über den Daten-Puffer. Diese Puffer-Einrichtung ist für den Zweiweg-Betrieb geeignet, d.h. über den Puffer kann eingeschrieben, aber auch ausgelesen werden. Der Daten-Puffer ist die Schnitt-Stelle zwischen dem internen Daten-Bus und dem externen Daten-Bus.
Bei einer Ausgabe-Operation wird der Inhalt des internen Daten-Busses zunächst in einen 8-Bit-Signal-Speicher geladen, der anschließend die Ausgangspuffer des Mikroprozessors ansteuert. Die Ausgangspuffer sind während einer Daten-Aufnahme und außerhalb einer Übertragungs-Operation ausgeschaltet.
Bei der Eingabe von Daten von dem externen zum internen Daten-Bus wird zu Beginn jedes internen Operationsschrittes der interne Bus geladen, aber nicht, wenn ein Übertragungsschritt erfolgt.

7.3. Arbeitszyklus des 8080

Der Mikroprozessor benötigt zum Laden, transportieren oder operieren eine bestimmte Zeit, die sogenannte Arbeitszeit oder den Arbeitszyklus. Ein Befehlszyklus wird definiert als die Zeit, die erforderlich ist, einen Befehl abzurufen und auszuführen. Während des Abrufes wird ein aus 1, 2 oder 3 Bytes bestehender Befehl aus dem Speicher geholt und in das Befehlsregister des Mikroprozessors übertragen. Während dieser Übertragungsphase des Befehlszyklus wird der Befehl decodiert. Jeder Befehlszyklus (instruction cycle) besteht aus einem oder mehreren Operationszyklen. Im allgemeinen haben wir 1 bis 5 Operationszyklen. Ein Operationszyklus (machine cycle) ist jedesmal erforderlich, wenn der 8080 den Speicher oder einen I/O-Baustein adressiert.

Der Mikroprozessor 8080 arbeitet synchron, d.h. der Arbeitsablauf erfolgt zyklisch. Der 8080 holt sich einen Befehl ein, führt die erforderliche Operation aus, holt sich den nächsten Befehl, führt diesen aus usw. Dieses nacheinander arbeiten, Daten und Befehle holen, verarbeiten, erfordert von dem Mikroprozessor einen genauen Zeitablauf mit einer zeitlichen Kontrolle, das sogenannte „timing".
Den Takt für das Timing liefert uns ein Quarz-Oszillator. Mit der Taktfrequenz können wir die einzelnen, zeitlichen Bezugssignale für den Funktionsablauf des Programmes erstellen.
Das Einholen und Ausführen der einzelnen Befehle wird als Befehls-Zyklus bezeichnet. Die aktiven Zeiteinheiten eines Zyklus bezeichnen wir als Operationsschritte (states). Die Taktperiode ist der Zeitintervall zwischen zwei aufeinander folgenden Impulsen des Taktgebers. Für das gesamte Timing eines Operationsschrittes sind jeweils eine oder mehrere Taktperioden erforderlich, d.h. ein Befehlszyklus besteht fast immer aus mehreren Operationsschritten.

Der erste Operationsschritt eines Befehlszyklus hat die Aufgabe, den nächsten Befehl abzurufen. Der Mikroprozessor gibt dazu ein Startsignal ab und das Adressenwort im Programmzähler wird zu einem bestimmten Speicher übertragen. Dieser reagiert, indem er das dementsprechende Befehlswort an den Mikroprozessor übermittelt. Das erste Wort des Befehls kann daher im Befehlsregister gespeichert werden.

Besteht der Befehl aus einem Zwei- oder Drei-Byte-Wort, so sind weitere Abruf-Schritte (FETCH) notwendig, und zwar solange, bis die gesamten Befehlsbytes eingelesen worden sind. Ist der gesamte Befehl im Mikroprozessor, so kann der Programmzähler in Vorbereitung des nächsten Befehlsabrufes um eine Eins erhöht und der eingeholte Befehl im Befehlsdecoder decodiert werden. Die im Befehl enthaltene Operation wird in den weiteren Operationsschritten des Befehlszyklus aufgeführt. Der Befehl könnte z.B. eine Anweisung für das Lesen oder Einschreiben eines Speichers enthalten, für eine Dateneingabe oder -ausgabe, ebenso für interne Mikroprozessoren-Operationen, z.B. Datenübertragung von einem Register in ein anderes oder in eine Register-Addition.

Die Befehlsabfrage ist ein spezieller Lesevorgang in einem Speicher, bei dem ein Befehl aus dem Befehlsspeicher in das Befehlsregister des 8080 übertragen wird. Der abgefragte oder abgerufene Befehl kann den Mikroprozessor anschließend veranlassen, daß die Daten aus dem Datenspeicher direkt in ihn hineingelesen werden. Hierzu gibt der Mikroprozessor wieder ein Lesesignal ab und adressiert den Speicher. Der Speicher reagiert durch die Übersendung des angeforderten Datenwortes. Dieses Datenwort wird von dem Mikroprozessor entweder in den Akkumulator oder in einem der allgemeinen Register gespeichert. Die Speicherung kann aber nicht im Befehlsregister erfolgen.

Soll in einem Speicher eine Information eingeschrieben, also gespeichert werden, so verläuft die Operation ähnlich wie beim Lesevorgang, abgesehen vom Datenfluß. Der Mikroprozessor gibt ein Schreibsignal ab, adressiert den Speicher mit einem Adreßwort und gibt anschließend das Datenwort zum Einschreiben in den adressierten Speicherplatz frei.

Für die Durchführung eines Befehlszyklus benötigt man eine bestimmte Zeit, die wesentlich aber von der Art des Befehls abhängt. Einige Befehle erfordern nur einen Operationszyklus, die zum Abrufen eines Befehls benötigt werden. Andere Befehle erfordern weitere Operationszyklen, um Daten in den Speicher oder I/O-Einrichtungen zu schreiben oder zu lesen.
Die maximale Geschwindigkeit, mit der ein Zyklus ablaufen kann, wird durch die Zugriffszeit, der

„access time", des Speichers begrenzt. Gibt der Mikroprozessor an den Speicher eine Adresse, so kann der Mikroprozessor erst dann wieder weiterarbeiten, wenn der Speicher reagiert hat. Die meisten Speicher sind zwar schneller als es der Ablaufzyklus erfordert, aber es gibt auch Speichertypen, die das adressierte Wort nicht innerhalb der Minimalzeit liefern können, die sich aus der Taktperiode des Mikroprozessors ergibt. Der Mikroprozessor hat daher einen speziellen Ausgang zum Synchronisieren des Speichers. Der Mikroprozessor gibt auf den Speicher den Wartezustand (WAIT). Wenn der Speicher ein Lese- oder Schreib-Signal zur Freigabe erhält, dann veranlaßt die Bereit-Leitung (READY) mit einem Signal den Stillstand des Mikroprozessors. Der Mikroprozessor bleibt stehen und der Speicher kann arbeiten. Hierzu sind die Speicher mit „enable"-Eingängen ausgerüstet. Erst wenn der Speicher genügend Zeit hatte um zu reagieren, gibt der Mikroprozessor die READY-Leitung wieder frei und der Befehlszyklus läuft weiter ab.

Die Ein- und Ausgabe von Daten verläuft ähnlich wie der Schreib/Lese-Vorgang bei den Speichern. Bei den Ein/Ausgabe-Einheiten wird nicht der Speicherplatz adressiert, sondern die entsprechende Einrichtung. Der Mikroprozessor erteilt über die Adressen-Leitungen den einzelnen Geräten zu Senden oder zu Empfangen.

Bei der 8080 können wir eine Programm-Unterbrechung in das Programm, den „interrupt" eingeben. Dadurch wird die Leistungsfähigkeit des Systems wesentlich erhöht. Arbeitet der Mikroprozessor an einer großen Datenmenge und sollen die Ergebnisse an einen Drucker abgegeben werden, so arbeitet das System im „interrupt"-Betrieb. Der Mikroprozessor kann zwar ein Datenwort innerhalb einer einzigen Zykluszeit ausgeben, aber ein Drucker benötigt wesentlich längere Arbeitszeiten, um seine Zeichen zu drucken. Dadurch müßte das System solange in Wartestellung gehen, bis der Drucker das nächste Datenwort verarbeiten kann. Haben wir aber die Möglichkeit eines Interrupts im Programm, so sendet das System ein Datenwort aus, kehrt anschließend sofort auf sein Programm zurück und arbeitet wieder.
Immer wenn der Drucker ein neues Datenwort empfangen, also verarbeiten kann, sendet er eine Unterbrechungsaufforderung an den Mikroprozessor. Der Mikroprozessor nimmt den Unterbrechungs-Befehl auf und unterbricht das Programm. Er springt in ein Unterprogramm und holt sich hieraus das Datenwort für den Drucker. Nach der Ausgabe des Datenwortes fährt der Mikroprozessor im Hauptprogramm wieder weiter.
Hierbei müssen wir beachten, daß dieser Interrupt-Vorgang im Prinzip einer CALL-Operation sehr ähnlich ist, abgesehen von der Unterbrechung mit dem Sprung in ein Unterprogramm. Diesen Interrupt können wir nur hardwaremäßig extern erzeugen.
Wir können durch die Hardware auch komplexere Unterbrechungen durchführen, bei denen mehrere zur Ausgabe von Interrupt-Befehlen berechtigte Einrichtungen mit verschiedenen Prioritätsgraden an demselben Prozessor angeschlossen sind. Ablaufsteuerungen mit Interrupt-Möglichkeiten sind eine der wichtigsten Eigenschaften und Eigenheiten bei Mikroprozeß-Systemen. Hier können wir das System optimal ausnützen und erreichen eine hohe Wirtschaftlichkeit durch den hohen Durchsatz in der Datenverarbeitung.

Die Leitungsfähigkeit eines Mikroprozessor-Systems kann noch durch den Einsatz eines Halt-Zustandes (HOLD) erhöht werden, d.h. durch ein geeignetes Signal kann das System angehalten werden. Dadurch ist ein „direkter Speicherzugriff" DMA möglich.

Bei der normalen Betriebsart überwacht der Mikroprozessor seine Ein/Ausgabe der Daten selbst. Informationen, die in den Speicher eingegeben werden sollen, können von einem entsprechenden Speicher zunächst zum Mikroprozessor übertragen und dann in einen vorgesehenen Speicherplatz gespeichert werden. In ähnlicher Arbeitsweise können Informationen vom Speicher über den Mikroprozessor zu einer Ausgabeeinheit transportiert werden.
Einige Peripheriegeräte können die Übertragung von Daten und Informationen zum oder vom Speicher selbst schneller durchführen, als der Mikroprozessor in der Lage ist. Bei der Übertragung von größeren Datenmengen, läßt sich der Durchsatz des Systems dadurch wesentlich steigern, wenn die Daten direkt von der Eingabe in den Speicher übermittelt werden. Der Mikroprozessor muß für diese Zeit der Datenübermittlung seine Operation kurzzeitig unterbrechen, um eventuelle Fehler zu vermeiden, die sich durch die gleichzeitige Adressierung des Speichers von Mikroprozessor und Peripheriegeräten ergeben könnten. Aus diesem Grund hat der 8080 einen Halt-Eingang (HOLD).

Jeder Operationszyklus besteht aus mehreren Operationsschritten (states). Ein Operationsschritt ist die kleinste Einheit in einem Programmablauf und wird definiert als das Zeitintervall zwischen zwei aufeinander folgenden, in positiver Richtung verlaufenden Übergänge des $\emptyset_1$-Taktimpulses. Der 8080 wird von einem 2-Phasen-Takt angesteuert. Alle Ablaufe sind daher zeitlich auf die Periode dieses Taktes bezogen. Die sich nicht überlappenden Taktimpulse $\emptyset_1$ und $\emptyset_2$ werden von einer externen Schaltung erzeugt. Der $\emptyset_1$-Takt teilt jeden Operationszyklus in einzelne Operationsschritte auf. Die zeitliche Steuer-Logik des 8080 leitet schon den Taktsignalen einen Synchronisationsimpuls (SYNC) ab, der den Anfang eines jeden Operationszyklus festlegt. Der SYNC-Impuls wird durch den $\emptyset_2$-Übergang von L nach H getriggert. Bild 7.3.1 zeigt den Ablauf für die beiden Taktimpulse und dem Synchronisationsimpuls.

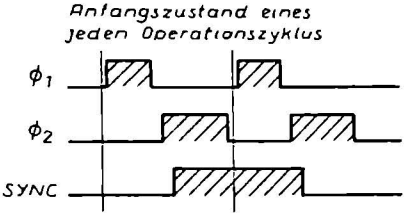


Bild 7.3.1 Zeitlicher Ablauf für die Taktimpulse und dem Synchronisationsimpuls

Drei Ausnahmen sind bei den Operationszyklen vorhanden. Es sind die Befehle „WAIT" für Warten, „HLDA" für Haltesteuerung und „HLTA" für das Quittungs-Signal des HALT-Befehles. Die drei Schritte sind von äußeren Bedingungen abhängig und häufig von unbestimmter Dauer. Aber auch diese drei Operationsschritte müssen wir mit den Taktsignalen synchronisieren. Die Dauer aller Operationsschritte ist deshalb immer ein ganzzahliges Vielfaches der Taktsignale.

7.4. Befehlszyklus der 8080

Der 8080 ist so aufgebaut, daß er nur ein Byte pro Operationszyklus übertragen kann. Wenn also der Abruf auf die Ausführung eines Befehls zwei Speicherzugriffe erfordert, muß der Befehlszyklus für den betreffenden Befehl aus zwei Operationszyklen bestehen. Wir benötigen für jeden Befehlszyklus mindestens eine Adressierung des Speichers, nämlich zur Abfrage des Befehls. Der erste Operationszyklus eines Befehlszyklus ist deshalb in jedem Fall ein Befehlsabruf, ein „FETCH". Der weitere Verlauf hängt von der Art des abgerufenen Befehls ab. Hierzu einige Beispiele:

Der Befehl „addiere Register (ADDr)" ist ein Befehl, zu dessen Durchführung wir nur einen einzigen Operationszyklus (Abruf) benötigen. Mit einem Ein-Byte-Befehl kann der Inhalt eines der 6 allgemeinen Register des 8080 zu den im Akkumulator befindlichen Daten hinzuaddiert werden. Da alle zur Ausführung des Befehls notwendigen Informationen im 8-Bit-Befehlscode enthalten sind, ist nur eine einzige Speicherabfrage erforderlich. Hierfür sind insgesamt drei Operationsschritte notwendig. Ein weiterer Schritt wird für die Durchführung der Addition benötigt. Der gesamte Befehlszyklus erfordert in diesem Fall also nur einen einzigen Operationszyklus, der aber aus vier Operationsschritten besteht und vier Perioden des externen Taktes beansprucht.

Soll der Inhalt eines Speicherplatzes zum Inhalt des Akkumulators addiert werden (ADD M), so ist die Aufgabe dem vorherigen Beispiel sehr ähnlich. Es wird nur ein weiterer Operationszyklus zum Einholen der Daten benötigt. Es ergibt sich dann folgender Ablauf: Zuerst empfängt der Mikroprozessor vom Speicher das von seinem Befehlszähler adressierte Ein-Byte-Befehlswort. Hierzu werden drei Operationsschritte benötigt. Dieses während des Abrufzyklus eingeholte 8-Bit-Befehlswort wird im Befehlregister

gespeichert und dazu verwendet, die weiteren Abläufe in dem betreffenden Befehlszyklus zu steuern. Als nächstes sendet der Mikroprozessor als Adresse den Inhalt seiner H- und L-Register aus. Das 8-Bit-Datenwort, das während dieses Speicher-Lesezykluses zurückkommt, wird in einem Zwischenspeicher-Register des 8080 aufgehoben. Hierfür werden drei weitere Taktperioden (Operationsschritte) benötigt. Im 7. und damit letzten Operationsschritt wird der Inhalt des Zwischenspeicher-Registers zu dem des Akkumulators addiert. Somit waren zur Durchführung des „ADD M"-Befehles insgesamt zwei Operationszyklen mit sieben Operationsschritten erforderlich.

Der Befehl „SHLD" (Registerpaar unter Adresse ADR und (ADR+1)abspeichern) ist einer der längsten Befehle in dem Mikroprozessor 8080 und besteht aus fünf Operationszyklen. Während eines SHLD-Befehlszyklus werden die Inhalte der H- und L-Register des Mikroprozessors in zwei aufeinander folgende Speicherplätze gegeben. Ihre Adressen werden durch zwei Adressen-Bytes angegeben, die sich direkt an die Adressen der dazugehörigen Operationscode-Bytes anschließen. Dadurch ergibt sich für uns folgender Programmablauf:

a) Ein Befehls-Abrufzyklus, der aus vier Operationsschritten besteht. Während der ersten drei Schritte dieses Zyklus holt der Mikroprozessor den von seinem Befehls-Zähler angezeigten Befehl ein. Der Inhalt des Befehlszählers wird anschließend um eine Eins erhöht. Der vierte Operationsschritt wird für die interne Befehlsdecodierung benötigt.
b) Ein Speicher-Lesezyklus, der aus drei Operationsschritten besteht. In diesem Zyklus wird das vom Befehlszähler angezeigte Byte vom Speicher abgerufen und im Z-Register des Mikroprozessors gespeichert. Der Inhalt des Befehls-Zählers wird nochmal um eine Eins erhöht.
c) Ein weiterer Speicher-Lesezyklus, der aus drei Operationsschritten besteht, in denen das vom Befehls-Zähler des Mikroprozessors angezeigte Byte aus dem Speicher gelesen und in das W-Register übertragen wird. Der Inhalt des Befehls-Zählers wird erneut um eine Eins angehoben, um damit die Vorbereitung für den nächsten Befehlsabruf einzuleiten.
d) Ein Speicher-Schreibzyklus aus drei Operationsschritten, in dem der Inhalt des L-Registers in eine Speicherstelle geschrieben wird, deren Adresse durch den Inhalt der Register W und Z angegeben wird. Der nächste Operationsschritt, der nach dieser Übertragung folgt, erhöht die im Registerpaar W und Z enthaltene Adresse, so daß diese den nächsten Speicherplatz anzeigt, der die Daten erhalten soll.
e) Ein Speicher-Schreibzyklus mit drei Operationsschritten, in dem der Inhalt des H-Registers zu der neuen vom Registerpaar W und Z angezeigten Speicherstelle übertragen wird.

Wenn wir uns den SHLD-Befehl als ganzes betrachten, dann benötigen wir für die 5 Operationszyklen insgesamt 16 Operationsschritte.

Der Mikroprozessor 8080 ist so ausgelegt worden, daß wir nie mehr als 5 Operationszyklen für einen Befehlszyklus benötigen.

Es gibt zehn verschiedene Arten von Operationszyklen, die innerhalb eines Befehlszyklus auftreten können:

a) FETCH, M1 = Befehlsabruf
b) MEMORY READ = Speicher auslesen
c) MEMORY WRITE = Speicher einschreiben
d) STACK READ = Stapelspeicher lesen
e) STACK WRITE = Stapelspeicher einschreiben
f) INPUT = Eingabe
g) OUTPUT = Ausgabe
i) INTERRUPT = Unterbrechung
j) HALT = Halt
k) HALT – INTERRUPT = Halt und Interrupt

Die Anzahl der Operationszyklen, die sich innerhalb eines Befehlszyklus ergeben, hängen von der Art des Befehles ab. Der erste Operationszyklus eines Befehlszyklus ist immer ein Befehlsabruf (FETCH). Der Mikroprozessor kennzeichnet jeden Operationszyklus, indem er während des ersten Schrittes ein 8-Bit-Zustandswort ausgibt. Die jeweilige Zustands-Information erscheint während des SYNC-Intervalls (Synchronisation) auf den Daten-Bus D_0 bis D_7. Sie wird in den Flip-Flops gespeichert und zur Ableitung von Steuersignalen für die externen Schaltungen verwendet.
Zustandssignale dienen vor allem zur Steuerung externer Einrichtungen. Dabei ist zu beachten, daß bestimmte Operationszyklen durch ein einzelnes Status-Bit klar gekennzeichnet sind, andere jedoch nicht. Das M1-Zustands-Bit D_6 kennzeichnet sofort eindeutig einen Abrufzyklus. Andererseits wird ein Stapel-Lesevorgang erst durch das Zusammentreffen der Signale STACK und MEMR angezeigt.

STACK = Bit auf dem Daten-Bus D_2 zeigt an, daß auf dem Adreß-Bus die Adresse des Stapelzeigers liegt.

MEMR = Bit auf dem Daten-Bus D_7 zeigt an, daß der Daten-Bus die Speicher-Lese-Daten aufnimmt.

Signale zur Kennzeichnung des Operationszyklus sind auch wichtig in der Prüf- und Fehlerbeseitigungsphase bei einer Systementwicklung.
In der 8080 sind 8 Daten-Leitungen vorhanden. Dadurch können wir folgende Zustands-Symbole übertragen.

D_0 für INTA = Bit auf diesem Daten-Bus D_0 zeigt das Quittungssignal für die Unterbrechungs-Anforderung (interrupt) an. Es wird auch verwendet, um einen RESTART-Befehl auf den Daten-Bus zu leiten, wenn DBIN aktiv wird.

D_1 für $\overline{WO}$ = Bit auf diesem Daten-Bus D_1 zeigt an, daß die Operation im gegenwärtigen Operationszyklus entweder eine „Speicher-Schreib"- oder eine „Ausgabe"-Operation ($\overline{WO}$ = L) ist. Andernfalls wird eine „Speicher-Lese"- oder eine „Eingabe"-Operation ausgeführt.

D_2 für STACK = Bit auf diesem Daten-Bus D_2 zeigt an, daß auf dem Adreß-Bus die Adresse des Stapelspeichers liegt.

D_3 für HLTA = Bit auf diesem Daten-Bus D_3 zeigt an, ob ein Quittungssignal für den HALT-Befehl vorliegt.

D_4 für OUT = Bit auf diesem Daten-Bus D_4 zeigt an, daß auf dem Adreß-Bus die Adresse eines Ausgabekanals liegt, und daß der Daten-Bus die Ausgabedaten enthalten wird, sobald der WR aktiv wird.

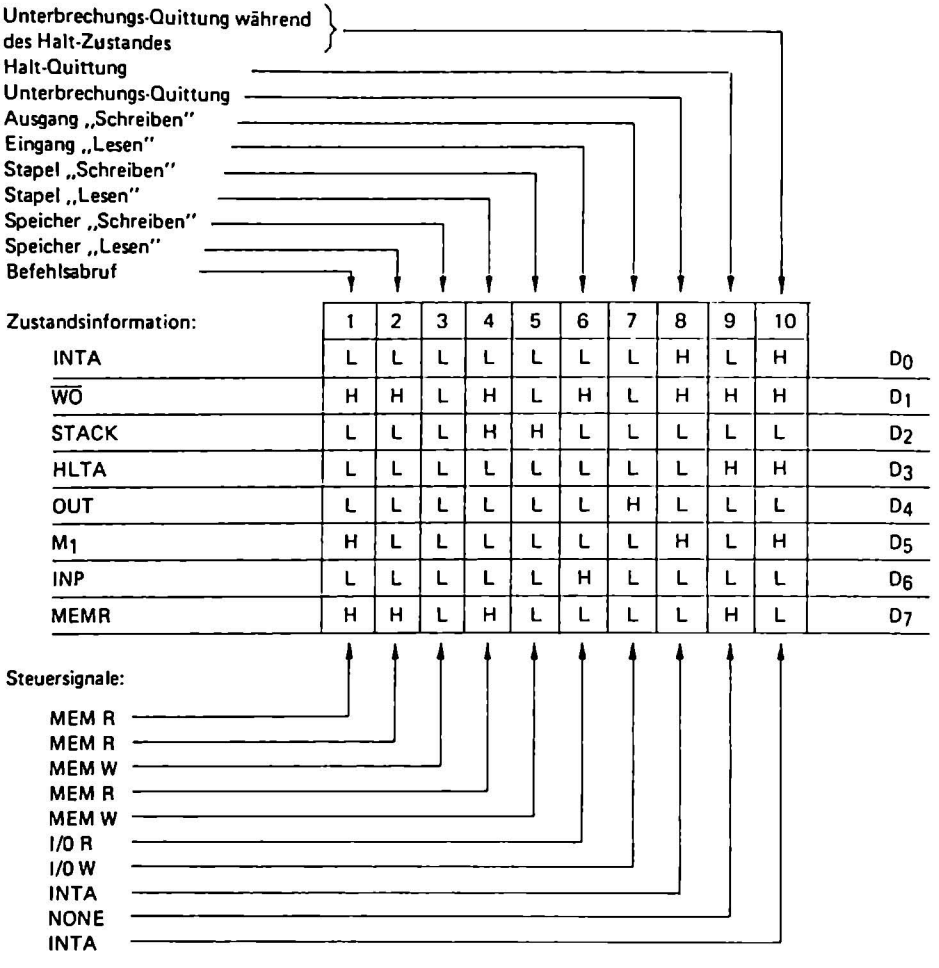
D_5 für M_1 = Bit auf diesem Daten-Bus D_5 zeigt an, daß sich der 8080 in einem Abruf-Zyklus zum Einholen des ersten Befehls-Byte befindet.

D_6 für INP = Bit auf diesem Daten-Bus D_6 zeigt an, daß auf dem Adreß-Bus die Adresse einer Eingabeeinrichtung liegt und die Eingabedaten auf den Daten-Bus gegeben werden, sobald DBIN aktiv wird.

D_7 für MEMR = Bit auf diesem Daten-Bus D_7 zeigt an, daß der Daten-Bus die Speicher-Lese-Daten aufnimmt.

Mit den drei Zustands-Symbolen INTA, INP und MEMR können wir den Datenfluß der 8080 steuern.

Art des Arbeitsablaufes:



7.5. Reihenfolge der Operationsschritte

Jeder Operationszyklus innerhalb eines Befehlszyklus besteht aus drei bis fünf aktiven Operationsschritten, den sogenannten „states", die wir mit T_1, T_2, T_3, T_4, T_5 oder mit T_W bezeichnen. Die Anzahl der Operationsschritte hängt von dem auszuführenden Befehl ab und von dem speziellen Operationszyklus. Der 8080 führt in jedem Operationszyklus mindestens drei Operationsschritte durch, wobei jeder Schritt durch zwei aufeinander folgende Übergänge des $\emptyset_1$-Taktes von L nach H begrenzt ist. In Bild 7.5.1 ist der zeitliche Ablauf eines typischen Abruf-Operationszyklus gezeigt. Vorgänge, die sich innerhalb eines Operationsschrittes abspielen, sind zeitlich auf die Signal-Übergänge der $\emptyset_1$- und $\emptyset_2$-Taktimpulse bezogen.

Das SYNC-Signal zum Synchronisieren kennzeichnet in jedem Operationszyklus den ersten Operationsschritt mit T_1. In dem Bild 7.5.1 ist das SYNC-Signal auf die führende Flanke des $\emptyset_2$-Taktes bezogen. Zwischen dem L-H-Übergang von dem $\emptyset_2$-Taktimpuls und der positiven Flanke des SYNC-Signales tritt eine Verzögerung ein. Eine entsprechende Verzögerung ergibt sich auch zwischen dem folgenden $\emptyset_2$-Impuls und der abfallenden Flanke des Synchronisations-Signals. Eine Zustands-Information wird während des gleichen Intervalls zwischen zwei $\emptyset_2$-Takten auf den Leitungen D_0 bis D_7 angezeigt. Ein- und Abschalten des Zustands-Signales wird ebenfalls durch den $\emptyset_2$-Takt angesteuert.

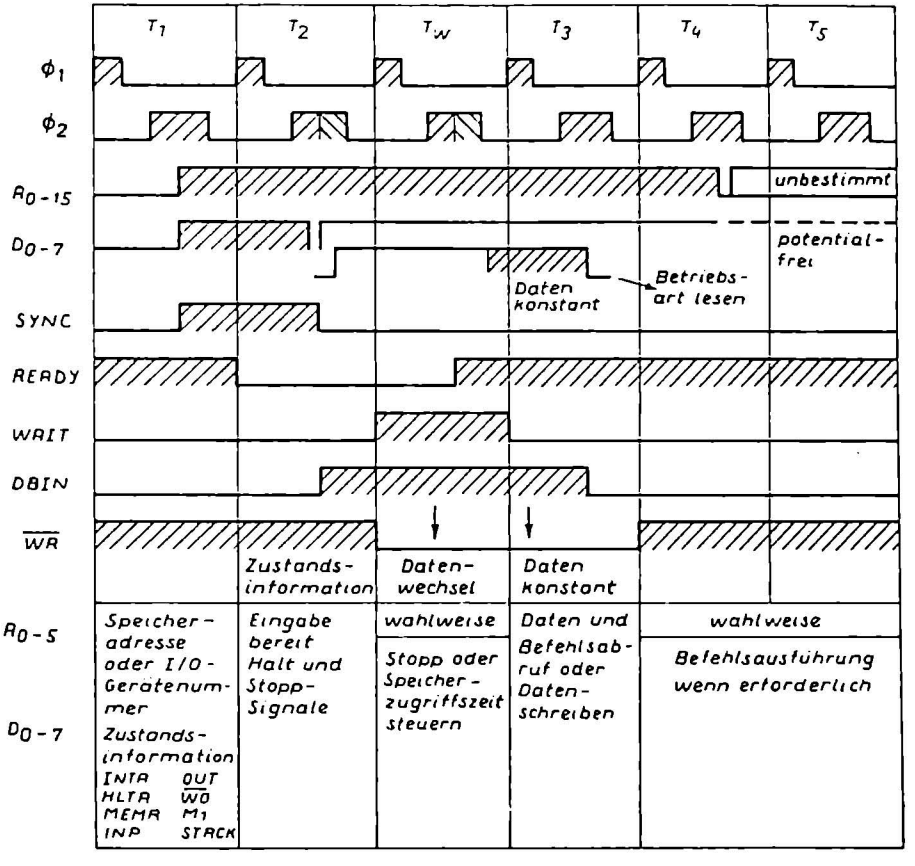


Bild 7.5.1 Zeitlicher Ablauf eines typischen Abruf-Operationszyklus

Die ansteigende $Ø_2$-Taktflanke während des ersten Operationsschrittes T_1 lädt auch die Adressenleitungen des Mikroprozessors A_0 bis A_{15}. Diese Signale sind innerhalb einer kurzen Verzögerungszeit nach dem $Ø_2$-Taktimpuls vorhanden und bleiben bis zum ersten $Ø_2$-Taktimpuls nach dem Operationsschritt T_3 erhalten. Hierzu erhält der Mikroprozessor etwas Zeit, die Daten zu lesen, die er vom Speicher erhalten hat.
Sobald der Mikroprozessor eine Adresse an den Speicher ausgesendet hat, hat dieser die Möglichkeit, einen Wartezustand (WAIT) anzufordern. Er kann diesen veranlassen, indem er die Bereits-Leitung (READY) des Mikroprozessors auf ein L-Signal schaltet und zwar vor einem READY SETUP-Intervall, das sich während des $Ø_2$-Taktes innerhalb eines Operationsschrittes T_2 oder T_W ergibt. Solange sich die READY-Leitung auf einem L-Signal befindet, bleibt der Mikroprozessor stehen. Er gibt hierzu dem Speicher eine Zeit, auf die Datenanforderung zu reagieren.
Der Mikroprozessor reagiert auf eine Warte-Anforderung, indem er am Ende des zweiten Operationsschrittes T_2 in Bild 7.5.1 den Operationsschritt TW einführt, anstatt direkt zum nächsten Operationsschritt T_3 überzugehen. Der Einstieg in den TW-Schritt wird vom 8080 durch ein Warte-Signal angezeigt, welches die Anforderungen, Lesen oder Schreiben, des Speichers bestätigt. Ein L-H-Übergang auf der Warte-Leitung wird von dem $Ø_1$-Taktimpuls getriggert und geht nach einer kurzen Verzögerungszeit später vor sich als der erfolgte Einstieg in den TW-Schritt.
Eine Warte-Periode kann von unbestimmter Dauer sein. Der 8080 bleibt im Wartezustand, bis seine Bereit-Leitung wieder ein H-Signal hat. Eine Bereit-Anzeige muß sich um ein bestimmtes Zeitintervall vor dem H-L-Flanke des $Ø_2$-Taktes einstellen, damit ein Verlassen des TW-Schrittes sichergestellt ist. Der Zyklus kann anschließend weiterarbeiten und beginnt mit dem nächsten L-H-Signal des $Ø_1$-Taktimpulses. Ein Warte-Intervall besteht daher aus einer ganzzahligen Anzahl von T_W-Schritten und trägt immer ein Vielfaches der Taktperiode.

7.6. Operationsschritte in der 8080

Im wesentlichen treten in der 8080 folgende Operationsschritte auf:

T_1: Eine Speicheradresse oder eine I/O-Kanal-Nummer wird auf den Adressen-Bus A_0 bis A_{15} und eine Status-Information auf den Daten-Bus D_0 bis D_7 gegeben.

T_2: Der 8080 prüft die Bereit- und HALT-Eingänge und untersucht, ob HALT-Befehle vorliegen.

T_W: Kann wahlweise für T_2 verwendet werden. Der 8080 geht in einen Wartezustand über, wenn sich das Bereit-Signal auf L-Signal befindet oder wenn ein HALT-Befehl ausgeführt worden ist.

T_3: Vom Daten-Bus wird ein Befehlsbyte (Abruf-Operationszyklus), Daten-Byte (Speicher- oder Stapel-Lesezyklus) oder ein Unterbrechungs-Befehl (Unterbrechungs-Zyklus) in die 8080 eingegeben, oder es wird ein Daten-Byte (Speicher- oder Stapel-Schreibzyklus oder Ausgabe-Zyklus) auf den Daten-Bus ausgegeben.

T_4: Schritt ist nur bei bestimmten Befehlen für interne Prozessorabläufe nötig.

T_5: Schritt ist nur bei bestimmten Befehlen für interne Prozessorabläufe nötig.

Die Vorgänge eines T_3-Schrittes werden durch den jeweiligen Operationszyklus bestimmt. In einem Abrufzyklus erkennt der Mikroprozessor die Daten auf seinem Daten-Bus als einen Befehl an. Während eines Speicher- oder Stapel-Lesevorganges werden die Daten auf diesem BUS als Datenwort verstanden. Während eines Speicher-Schreibzykluses gibt der Prozessor auf demselben BUS seine Daten aus. Bei einer I/O-Operation kann der Mikroprozessor entweder seine Daten aussenden oder andere Daten empfangen.

In Bild 7.6.1 ist der zeitliche Ablauf einer Daten-Eingabe-Operation gezeigt. Wie aus dem Impulsdiagramm ersichtlich ist, wird während dem Operationsschritt T_2 beim L-H-Impuls des $Ø_2$-Taktes die Zustands-Information von den Datenleitungen des Mikroprozessors entfernt, damit diese Leitungen für den Empfang von Daten bereit sind. Die dem Mikroprozessor angebotenen Daten müssen rechtzeitig anliegen. Die Daten müssen außerdem während des Daten-Halt-Intervalls im Anschluß an diesem Impuls anliegen. Die von einem Speicher- oder einer anderen Einrichtung auf die Datenleitungen gegebenen Daten werden während dem Operationsschritt T_3 gelesen. Während der Dateneingabe in den Mikroprozessor erzeugt dieser ein DBIN-Signal, welches dann extern zur Freigabe der Übertragung verwendet wird.

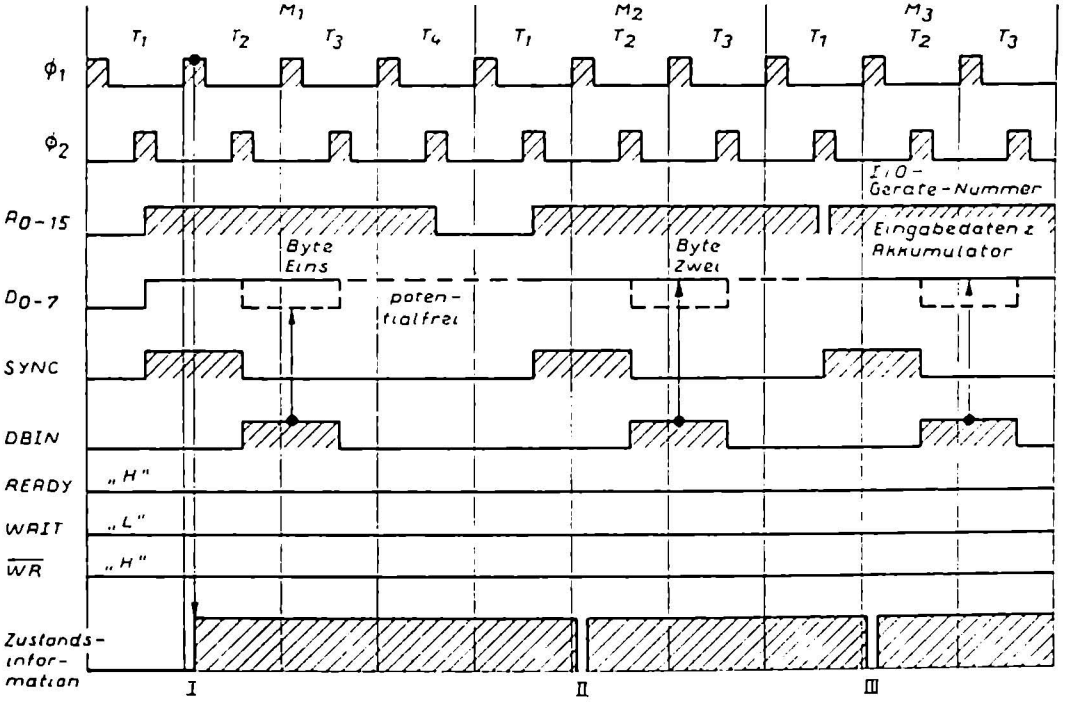


Bild 7.6.1 Zeitlicher Ablauf einer Quoten-Eingabe-Operation

			D0	D1	D2	D3	D4	D5	D6	D7
I = Befehlsabruf	⇒	MEM R - Steuersignal ⇒	L	H	L	L	L	H	L	H
II = Speicher „Lesen"	⇒	MEM R - Steuersignal ⇒	L	H	L	L	L	L	L	H
III = Eingang „Lesen"	⇒	I/O W - Steuersignal ⇒	L	H	L	L	L	L	H	L

Operationszyklen, bei denen DBIN verwendet wird, sind folgende Befehle:
Befehlsabruf, Speicher lesen, Stapelspeicher lesen oder Unterbrechung.
DBIN wird während des zweiten Operationsschrittes T_2 durch den L-H-Sprung von dem $\emptyset_2$-Takt erzeugt und durch die entsprechende Flanke von dem $\emptyset_2$-Takt während des dritten Operationsschrittes T_3 beendet. Wenn sich zwischen den beiden Operationsschritten T_2 und T_3 ein T_W-Operationsschritt ergibt, wird die DBIN um eine Taktperiode verlängert. Dasselbe gilt auch bei mehreren T_W-Operationsschritten. Pro T_W verlängern wir die Taktperioden.
In Bild 7.6.2 ist der zeitliche Ablauf eines Operationszyklus gezeigt, in dem der Mikroprozessor seine Daten aussendet. Die Ausgabedaten können entweder für ein Speicher- oder für ein Peripherie-Gerat bestimmt sein. Die L-H-Flanke des $\emptyset_2$-Taktes, während des zweiten Operationsschrittes T_2, ersetzt die Zustandsinformation auf dem Daten-Bus durch die Ausgabedaten für die externen Geräte. Dieser Vorgang findet während der Datenausgabe-Verzögerung statt, die der führenden Flanke des $\emptyset_2$-Taktimpulses folgt.
Die Daten auf dem Bus bleiben während der übrigen Zeit des Operationszyklus stabil, bis sie durch neue Zustandsinformationen im folgenden Operationsschritt T_1 ersetzt werden. Wir müssen nur beachten, daß für die vollständige Durchführung eines Ausgabe-Operationszyklus ein BEREIT-Befehl oder Signal benötigt wird. Wenn dieses Signal nicht vorhanden ist, geht der Mikrprozessor im Anschluß an den zweiten Operationsschritt T_2 in den T_W-Warte-Zustand über. Die an den Ausgangsleitungen befindlichen Daten bleiben während dieser Zeit stabil und der Operationszyklus wird nicht fortgesetzt, bis die Bereit-Leitung wieder auf H-Signal ist.

Der 8080 erzeugt ein $\overline{WR}$-Signal am Ausgang für die Synchronisation von den externen Abläufen, während der Datenausgabe-Operationszyklen. Die H-L-Flanke des $\overline{WR}$ wird von dem L-H-Sprung des ersten $\emptyset_1$-Taktimpulses im Anschluß an den zweiten Operationsschritt T_2 abgeleitet und tritt mit einer kurzen Verzögerung danach auf. Der $\overline{WR}$ bleibt auf L-Signal, bis er durch die L-H-Flanke des $\emptyset_1$-Taktes während des auf dem dritten Operationsschritt T_3 folgenden Operationsschrittes wieder getriggert wird. Wir müssen hierbei beachten, daß zwischen den Operationsschritten T_2 und T_3 des Ausgabe-Zyklus auftretende T_W-Zustände, also Warte-Zustände die $\overline{WR}$-Periode verlängern und zwar in etwa wie bei DBIN während der Daten-Eingabeoperationen beeinflußt wird.

7.7. Ablauf bei Programmunterbrechungen

Mit dem 8080 können wir externe Unterbrechungs-Anforderungen von Peripherie-Geräten durchführen. Ein Peripherie-Gerät kann eine Unterbrechung veranlassen, wenn sie auf die Unterbrechungs-Leitung (INT) ein H-Signal gibt.
Der Unterbrechungs-Eingang (INT) ist asynchron. Eine Unterbrechungsanforderung darf deshalb jederzeit während eines beliebigen Befehlszyklus gestellt werden. Eine interne Logik im Mikroprozessor taktet das externe Anforderungssignal, so daß stets die richtige zeitliche Zuordnung zum Taktsystem gegeben ist. In Bild 7.7.1 ist ein Impulsdiagramm für Programmunterbrechungen gezeigt. Wie das Impulsdiagramm zeigt, setzt eine Unterbrechungs-Anforderung (INT), die zu einem Zeitpunkt ankommt, an dem die Unterbrechungs-Freigabe-Leitung (INTE) auf H-Signal ist, in der Verbindung mit dem $\emptyset_2$-Taktsignal ein internes Unterbrechungs-Flip-Flop. Dieser Vorgang des Flip-Flop spielt sich während des letzten Operationsschrittes im Befehlszyklus ab, in dem die Anforderung auftritt. Damit ist sichergestellt, daß ein gerade in Bearbeitung befindlicher Befehl vor der Berücksichtigung der Unterbrechung vollständig ausgeführt wird.

In vieler Hinsicht ist der Unterbrechungs-Operationszyklus, der auf eine freigegebene Unterbrechungs-Anforderung folgt, einem gewöhnlichen Abrufzyklus ähnlich. Das M 1-Zustands-Bit wird wie gewöhnlich während des SYNC-Intervalls übertragen. Es wird jedoch jetzt von einem INTA-Zustands-Bit auf der Daten-Bus-Leitung D_0 begleitet, das die externe Anforderung quittiert. Der Inhalt des Befehlszählers wird während des Operationsschrittes T_1 auf die Adreß-Leitungen des 8080 geschaltet. Der Zähler wird jedoch während des Unterbrechungs-Zyklus nicht weitergezählt, wie es sonst bei einem Programmablauf üblich ist. Auf diese Weise wird der Zustand des Befehlszählers vor der Unterbrechung beibehalten und die Daten im Zähler können nach dem Abschluß des Unterbrechungs-Befehls durch das Unterbrechungs-Programm wieder hergestellt werden. Es ist dabei die Aufgabe der Peripherie-Geräte, dafür

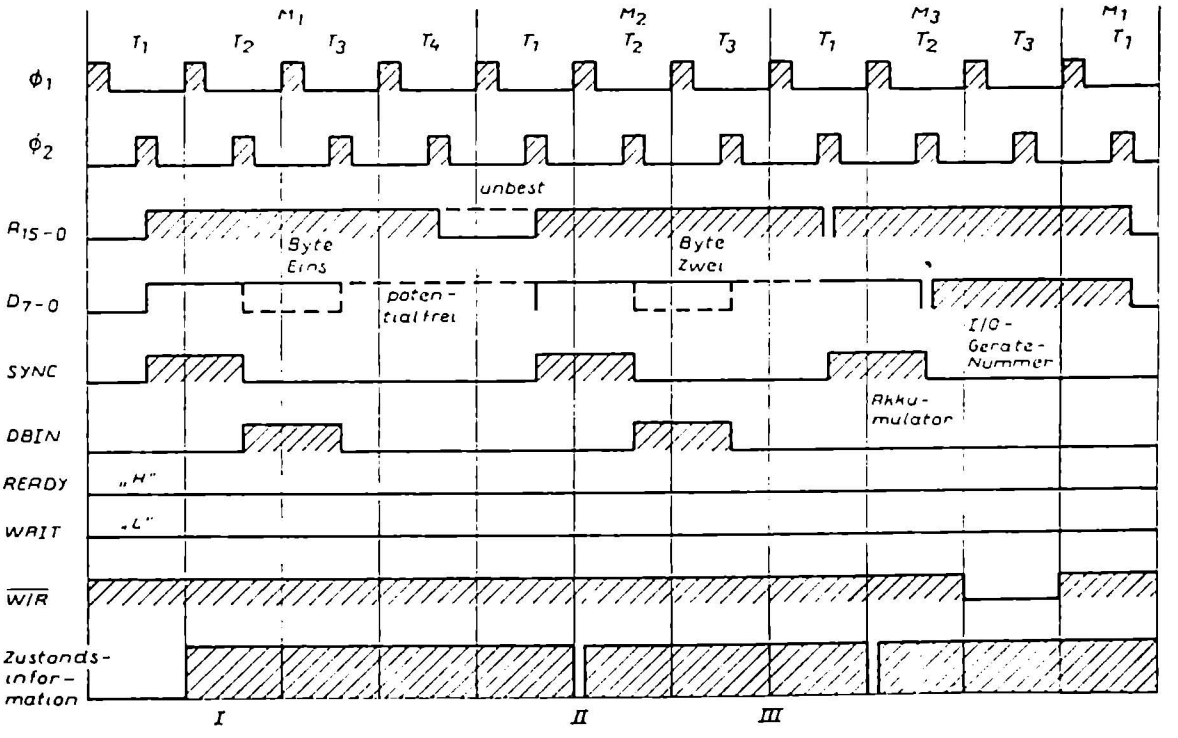


Bild 7.6.2 Zeitlicher Ablauf einer Datenausgabe-Operation

			D0	D1	D2	D3	D4	D5	D6	D7
I	= Befehlsabruf	⇒ MEM R - Steuersignal ⇒	L	H	L	L	L	H	L	H
II	= Speicher „Lesen"	⇒ MEM R - Steuersignal ⇒	L	H	L	L	L	L	L	H
III	= Ausgang „Schreiben"	⇒ I/O - Steuersignal ⇒	L	L	L	L	H	L	L	L

zu sorgen, daß eine 8-Bit-RESTART-Information während des dritten Operationsschrittes T_3 dem Daten-Bus des Mikroprozessors aufgeprägt wird. Dies bedeutet, daß der vom Speicher kommende Dateneingabe-Bus zeitweilig vom Daten-Bus des Mikroprozessors isoliert werden muß, damit die unterbrechende Schaltung auf den Daten-Bus des Mikroprozessors keinen Einfluß nehmen kann.

Der Befehlssatz des 8080 enthält einen besonderen 1-Byte-Aufrufbefehl, der die Bearbeitung von Unterbrechungen erleichtert. Wenn wir einen gewöhnlichen Programm-Aufruf-Befehl durchführen, so benötigen wir hierzu 3 Bytes. Dieser 1-Byte-Befehl ist der sogenannte RESTART-Befehl, der im Befehlssatz mit RST bezeichnet ist. Ein variables 3-Bit-Feld innerhalb des 8-Befehls-Codes des RST ermöglicht es der unterbrechenden Schaltung, einen Aufrufbefehl an einen von den acht festen Speicherplätzen zu richten. Die Dezimal-Adressen dieser reservierten Speicherplätze sind: 0, 8, 16, 24, 32, 40, 48 und 56. Diese Adressen dürfen verwendet werden, um den ersten Befehl, bzw. die ersten Befehle, für eine Operation zu speichern, die das jeweils unterbrechende Element erfordert. Da der RST ein Aufrufbefehl ist, wird bei seiner Beendigung der alte Inhalt des Befehlszählers im Stapelspeicher gespeichert.

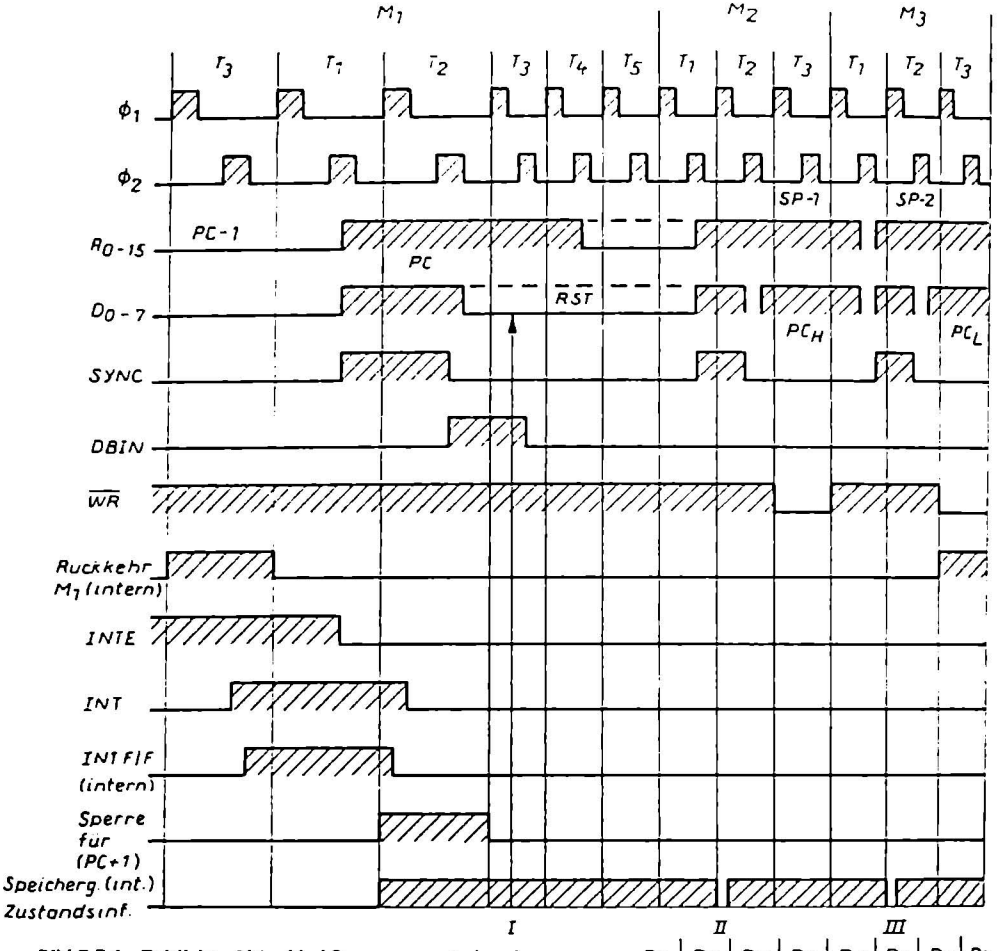


Bild 7.7.1 Zeitlicher Ablauf bei Programmunterbrechungen

		D0	D1	D2	D3	D4	D5	D6	D7
I = Unterbrechungs-Quittung ⇒ INTA - Steuersignal	⇒	H	H	L	L	L	H	L	L
II = Stapel „Schreiben" ⇒ MEM W - Steuersignal	⇒	L	L	H	L	L	L	L	L
III = Stapel „Schreiben" ⇒ MEM W - Steuersignal	⇒	L	L	H	L	L	L	L	L

7.8. Direkter Speicherzugriff bei der 8080

Wir können durch einen HOLD-Befehl den Mikroprozessor direkt anhalten. Dies geschieht, indem die den direkten Speicherzugriff angeforderten Peripherie-Daten vom Mikroprozessor ein entsprechendes Signal übermitteln, den HOLD-Befehl. Der Mikroprozessor reagiert auf eine solche Anforderung, indem er seine Adressen-Leitungen in den hochohmigen Zustand bringt und die Festlegung ihrer Potentiale dem entsprechenden Peripherie-Gerät überläßt. Dies ist nur mit Tri-State-Ausgängen möglich. Der Mikroprozessor quittiert das HOLD-Signal, indem er einen H-Pegel an seinen HOLD-Quittungs-Ausgang (HLDA) legt.
Wie der Unterbrechungs-Eingang ist auch der HOLD-Eingang intern synchronisiert. Ein HOLD-Signal muß vor dem HOLD-Vorbereitungsintervall, das der L-H-Flanke von dem $\emptyset_2$-Taktsignal vorausgeht, ein H-Signal angenommen haben.
Die Impulsdiagramme von Bild 7.8.1 und 7.8.2 zeigen den zeitlichen Ablauf von HOLD-Operationen. Wir müssen die Verzögerung zwischen der asynchronen HOLD-Anforderung und dem mit dem Takt verknüpften HOLD-Signal beachten. Wie das Diagramm in Bild 7.8.1 zeigt, wird das interne HOLD-Flip-Flop durch die Koinzidenz der READY-, HOLD- und des $\emptyset_2$-Taktimpulses gesetzt. Das Setzen dieses internen Flip-Flops ermöglicht den nächsten L-H-Sprung des $\emptyset_1$-Taktimpulses, den HOLD-Quittungs-Ausgang (HLDA) auf H-Signal zu schalten. Die Quittung der HOLD-Anforderung geht dem Floating der Adressen- und Daten-Leitungen des Mikroprozessors etwas voraus. Floating beim Tri-State-Ausgang bedeutet, der Ausgang wird hochohmig und kann damit Daten von anderen Ausgängen nicht mehr beeinflussen. Der Mikroprozessor quittiert einen Haltzustand (HOLD) zu Beginn von dem dritten Operationsschritt T_3, wenn ein Lese- oder ein Eingabe-Operationszyklus abläuft.
Dieser Ablauf ist in Bild 7.8.1 gezeigt.
Andernfalls wird die Quittung bis zum Beginn des folgenden Operationsschrittes T_3 zurückgestellt.

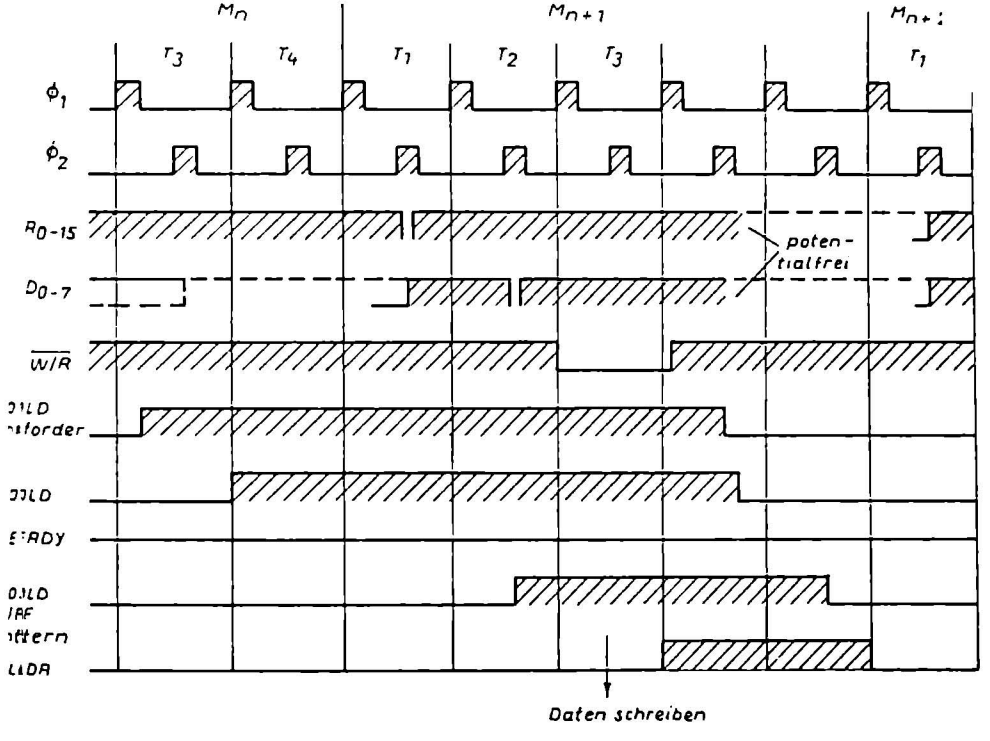


Bild 7.8.1 Zeitlicher Ablauf bei einem direkten Speicherzugriff im Lesevorgang

Bild 7.8.2 zeigt die HOLD-Operation, wenn geschrieben wird.
In beiden Fällen wird jedoch die HLDA-Leitung innerhalb einer bestimmten Zeit nach der L-H-Flanke des betreffenden $\emptyset_1$-Taktimpulses auf H-Signal geschaltet. Die Adressen- und die Daten-Leitungen gehen im Anschluß an die L-H-Flanke des nächsten $\emptyset_2$-Taktimpulses in den hochohmigen Zustand. Gehen alle Adressen- und Daten-Busse des 8080 in den hochohmigen Zustand, dann hat es momentan den Eindruck, wie wenn der Mikroprozessor stehen bleibt. Die internen Funktionen laufen aber weiter. Wenn z.B. eine HOLD-Anforderung im dritten Operationsschritt T_3 quittiert wird und wenn sich der Mikroprozessor gerade in einem Operationszyklus befindet, welcher vier oder mehrere Operationsschritte benötigt, dann durchläuft der Prozessor die Operationsschritte T_4 und T_5, bevor es zum Stillstand kommt. Der Ablauf wird nicht abgebrochen, bevor der Operationszyklus beendet ist. Auf diese Weise kann der interne Ablauf die externe DMA-Übertragung überbrücken.
Die Beendigung des HOLD-Zustandes im Mikroprozessor erfolgt mit einer ähnlichen Folge von Schritten wie seine Einführung. Eine HOLD-Anforderung wird asynchron beendet, wenn die externen Einrichtungen ihre Datenübertragung beendet haben. Der HLDA-Ausgang nimmt im Anschluß an die L-H-Flanke des nächsten $\emptyset_1$-Taktimpulses wieder ein L-Signal an. Der normale Ablauf wird damit fortgesetzt, und zwar mit dem Operationszyklus, der dem zuletzt ausgeführten Zyklus folgt.

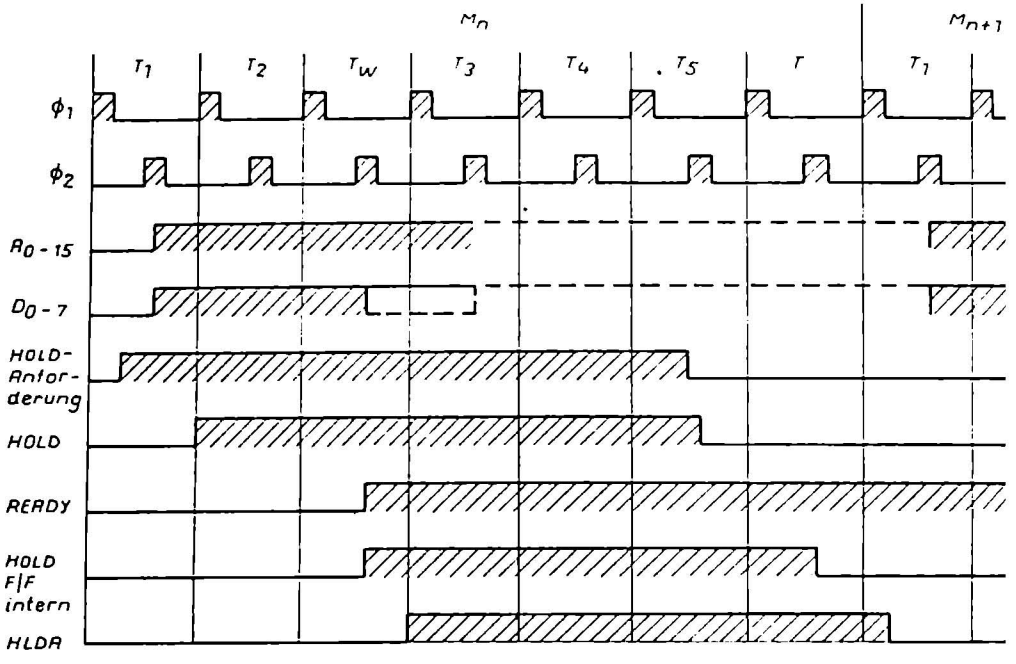


Bild 7.8.2 Zeitlicher Ablauf bei einem direkten Speicherzugriff im Lesevorgang

7.9. HALT-Ablauf bei der 8080

Wenn ein HALT-Befehl ausgeführt wird, geht der 8080 nach dem zweiten Operationsschritt des folgenden Operationszyklus in den HALT-Zustand (TWH) über. In Bild 7.9.1 ist das Impulsdiagramm gezeigt. Es gibt nur drei Möglichkeiten für den 8080, den HALT-Zustand zu verlassen:

a): Ein H-Signal auf der Rücksetz-Leitung (RESET) setzt den 8080 immer in den ersten Operationsschritt zurück. Das Rücksetz-Signal setzt auch den Befehlszähler zurück.

b): Eine HOLD-Eingabe bringt den 8080 in den HOLD-Zustand, wie zuvor beschrieben. Wenn die HOLD-Leitung ein L-Signal annimmt, geht der 8080 mit der L-H-Flanke des nächsten $\emptyset_1$-Taktimpulses in den HOLD-Zustand zurück.

c): Ein Unterbrechungs-Befehl (INT wird zu H-Signal, während INTE aktiviert ist) veranlaßt den 8080, den HALT-Zustand zu verlassen und mit der L-H-Flanke des nächsten Taktimpulses in den ersten Operationsschritt T_1 überzugehen.

Für die Möglichkeit ist es jedoch notwendig, daß beim Übergang in den HALT-Zustand das Unterbrechungs-Freigabe-Kennzeichen (INTE) gesetzt ist. Andernfalls kann der 8080 nur auf dem Wege über ein Rücksetz-Signal den HALT-Zustand verlassen.

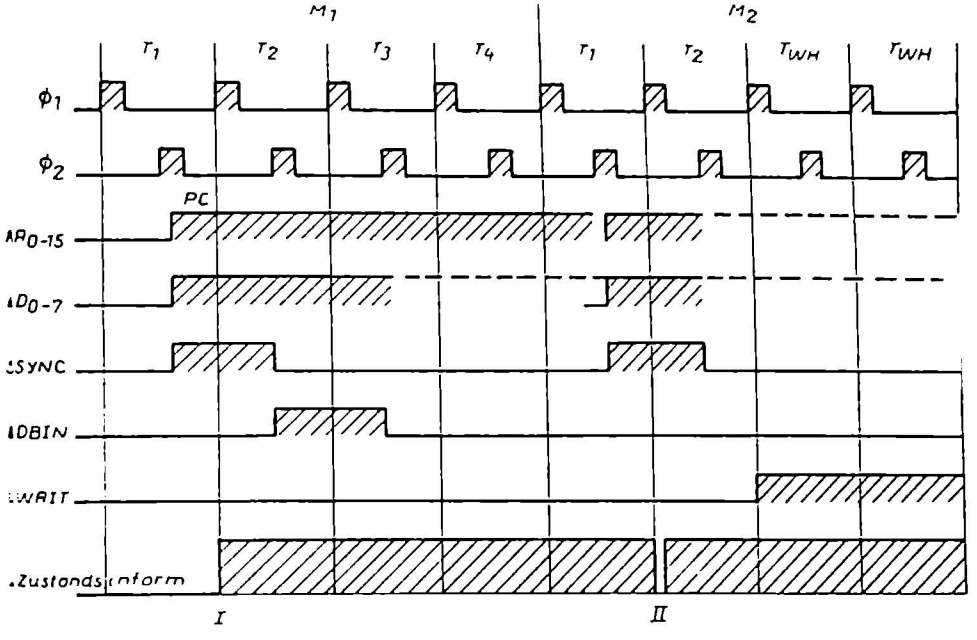


Bild 7.9 Zeitlicher Ablauf bei einem HALT der 8080

					D0	D1	D2	D3	D4	D5	D6	D7
I	= Befehl-Abruf	⇨	MEM R - Steuerbefehl	⇨	L	H	L	L	L	H	L	H
II	= Halt-Quittung	⇨	NONE - Steuerbefehl	⇨	L	H	L	H	L	L	L	H

7.10. Einschalten des 8080

Wird die Betriebsspannung in einem Mikroprozessor-System eingeschaltet, beginnt der Mikroprozessor zu arbeiten. Dabei wäre der Inhalt seines Befehlszählers, Stapelzeigers und der anderen Arbeitsregister vom Zufall abhängig. Der Mikroprozessor würde hier undefinierbar arbeiten. Aus diesem Grund ist es notwendig, den Einschaltmoment mit einem Rücksetzvorgang (RESET) einzuleiten.
Ein externes Rücksetz-Signal von mindestens drei Taktperioden Dauer setzt den Befehlszähler des Mikroprozessors auf Null. Die Ausführung des Programms beginnt dann im Anschluß an das Rücksetz-Signal mit dem Speicherplatz 0.
Systeme, in denen der Mikroprozessor auf ein besonderes Startsignal warten muß, speichern in den ersten zwei Speicherplätzen einen HALT-Befehl (EI und HLT). Zum Starten wird ein von Hand angelegter automatischer Unterbrechungsbefehl benutzt.
In anderen Systemen kann der Prozessor sofort mit der Verarbeitung der Anwenderprogramme beginnen. Es ist jedoch zu beachten, daß das Rücksetzen des Zustandskennzeichens oder die Arbeitsregister des Mikroprozessors, wie Akkumulator, Register oder Stapelzeiger, hiervon nicht beeinflußt werden. Der Inhalt dieser Register bleibt unbestimmt, bis sie durch das Programm in einen expliziten Anfangszustand versetzt werden.

Mikroprozessoren Teil 2

Frühjahr 1977
DM 19.80 ca. 150 Seiten
Best. Nr. 26

1. Technologie der Mikroprozessor-Speicherbausteine
1.1 Schreib-Lese-Speicher (RAM) statisch und dynamisch
1.2 Festwertspeicher (ROM)
1.3 Programmierbare Festwertspeicher (PROM)
1.4 Löschbare PROM (REPROM)
1.5 FIFO (first in, first out)
1.6 FILO (first in, last out)
1.7 Schieberegister statisch und dynamisch
1.8 MPR-Register (multiple port register)
1.9 ARL-Register (addressable register)
1.10 SAR-Register (successive approximations register)

2. Aufbau eines Mikroprozessor-Systems mit dem 8080
2.1 ROM-Schnittstellen
2.2 RAM-Schnittstellen
2.3 Schnittstellen für die Ein- und Ausgabe

3. Befehlssatz des 8080
3.1 Allgemeine Informationen über die Befehlsliste des 8080
3.2 Datenübertragungsoperation
3.3 Arithmetische Operationen
3.4 Logische Operationen
3.5 Register-Operationen
3.6 Sprungbefehle
3.7 Unterprogramme
3.8 Programmunterbrechungen
3.9 Pseudobefehle

Best.Nr.

1

TBB - Handbuch Band 1 , W. Hofacker

Transistor Berechnungs- und Bauanleitungshandbuch Band 1

Das Handbuch für jeden Elektroniker. Rechenbeispiele, Berechnungsgrundlagen, Bauanleitungen, Nomogramme, Tabellen und Vergleichslisten aus den wichtigsten Bereichen der Elektronik. Ein Buch zum Einarbeiten in die Elektronik. Ein Buch zum Nachschlagen. Grundlagen Digitaltechnik, Netzgeräte und Transformatorenberechnung, Berechnung von Multivibratoren, Schmitt Trigger u. v. a. Über 130 Seiten. **DM 19,80**

2

TBB - Handbuch Band 2 , W. Hofacker

Transistor Berechnungs- und Bauanleitungshandbuch Band 2

Dieses Buch ist die Fortsetzung des erfolgreichen Handbuches TBB-Handbuch Band 1. Ein Buch, das sich in der Hand des Praktikers bestens bewährt hat. Weitere neueste Schaltbeispiele und Berechnungsgrundlagen. Experimentier- und Versuchsbeschreibungen. Integrierte Spannungsregler, Wärmeableitung, Operationsverstärker Einführung, RC-Zeitglieder, Transistortester u. v. a. **DM 19,80**

3

Elektronik im Auto , H. Gebauer

Ein Buch für jeden technisch interessierten Autofahrer. Es zeigt Ihnen die vielen Möglichkeiten zur Verbesserung von Sicherheit, Leistung und Fahrkomfort in Ihrem Auto. Thyristorzündung, Drehzahlmesser, Beschleunigungsmesser, Geschwindigkeitswarner, Batterieladegerät u. v. a. Tips, genaue Beschreibungen, Bauanleitungen. **DM 9,8(**

4

IC - Handbuch , C. Lorenz

Handbuch für digitale und lineare integrierte Schaltungen.

Sensationelle Neuheit. Ein Handbuch für digitale und lineare Schaltkreise. Daten- und Auswahllisten, Vergleichslisten, Gehäuseformen, Grundlagen, Einführungsbeschreibungen, viele Schaltbeispiele, Bauanleitungen, Printvorlagen, u. v. a.
Alles über TTL-Technik, C MOS, MOS-Schaltungen, Uhren ICs, lineare Schaltungen, ein Chip-Rechner, integrierte NF-Verstärker u. v. a. **DM 19,80**

5

IC - Datenbuch , D. Steinbach

Daten- und Auswahllisten der gebräuchlichsten integrierten Schaltkreise. Digital und analog. Gerade bei ICs ist es wichtig die Anschlußfolgen genau zu kennen. Auf über 55 Seiten finden Sie: Die wichtigsten TTL-Schaltkreise, C-Mos Serie, lineare Schaltungen wie Operationsverstärker, Komparatoren, NF-Verstärker, Spannungsregler, Triggerschaltungen, Impulsgeber u. v. a. Weiterhin finden Sie eine C Mos-Vergleichsliste sowie Kurzdaten und logisches Verhalten dieser C MOS Elemente. Das IC-Datenbuch wird auch Ihnen ein unentbehrlicher Begleiter bei allen Arbeiten mit integrierten Schaltungen sein. **DM 9,80**

6

IC - Schaltungen , D. Steinbach

Hier finden Sie eine gelungene Zusammenstellung der wichtigsten Anwendungsbeispiele aus dem Bereich der integrierten Schaltungen. TTL – C MOS – Linear. Alle Schaltungen sind übersichtlich und klar dargestellt und mit einer kurzen, jedoch sehr genauen Beschreibung versehen. Viele Schaltungen sind Grundschaltungen, die man beim Umgang mit integrierten Schaltungen immer wieder benötigt. Tastenentprellung, Zähler, Impulsgeber, Codierer, Dekodierer, Datenübertragung, Serien-Parallel-Wandler, Digitalvoltmeter u. v. a. **DM 9,80**

7

Elektronik Schaltungen,
4. völlig neu überarbeitete Auflage, W. Hofacker

Die ideale Schaltungssammlung zum Basteln und Experimentieren. Schaltungen mit Operationsverstärkern, Spannungsreglern, TTL, C-MOS Schaltkreisen. MOS Uhr mit Wecker, elektronischer Würfel, Musik Synthisizer, Timer 555 Anwendungen, Experimentieranleitungen und viele andere hochinteressante Schaltbeispiele tlw. mit Printvorlage. 64 Seiten Inhalt. **DM 5,–**

8

IC - Bauanleitungs -Handbuch -IC -KIT , C. Lorenz

Ein Bauanleitungsbuch mit vielen hochinteressanten Bauanleitungen aus dem Bereich der LSI Schaltungstechnik. Schaltbeispiele mit Printvorlagen zum Selbstherstellen der Leiterplatten mit genauesten Beschreibungen. Hochaktuell und brandneu: Funktionsgenerator XR 2206, MOS-Uhr mit Wecker, Schlummerautomatik und programmierbarem Wecktongenerator, Sensortastenwahl, IC-Netzteil, Funktionsgenerator 8038 neuartige Transistorzündung, 35 W NF-Verstärker, Experimentieranleitung und Grundkurs über Flip Flops, Experimente mit Digitalschaltungen u. v. a. Zu allen Schaltungen finden Sie Platinvorlagen oder Sie können die Experimentierschaltungen auf der Experimentierplatine WH-1 g durchführen. Über 125 Seiten. **DM 19,80**

41

Experimentierplatine mit Sockel, Stecker und Füßen Typ WH-1g für 40,28,24,16 und 14 polige DIL -Gehäuse **DM 79,–**

9

Feldeffekttransistoren, C. Lorenz

Der Feldeffekttransistor (FET) gehört heute zu den interessantesten Bauteilen überhaupt. Wie man damit experimentiert, wie man seine Funktion versteht und wie man damit brauchbare und hochinteressante Schaltungen aufbauen kann, zeigt Ihnen dieses Buch. Grundlagen, Kennlinienfelder, Tabellen, Berechnungsgrundlagen, Rechenbeispiele, Anschlußbilder und eine Vergleichsliste für Feldeffekttransistoren bilden den Kern dieser umfangreichen Darstellung. Alles in allem finden Sie hier eine praxisnahe und komplette Arbeitsunterlage, mit der Sie im Beruf und auch im Hobby erfolgreich arbeiten können. Über 45 Seiten. **DM 5,–**

10

Elektronik und Radio, C. Lorenz
4. Auflage. Völlig neu bearbeitet und stark erweitert.

Eine Einführung in die Radiotechnik, wie man sie nicht alle Tage findet. Eine sehr geschickt gemachte Einführung mit vielen Schaltungen, Bauanleitungen und genauesten Funktionsbeschreibungen. Vom einfachen Diodenempfänger (Detektor) bis zu interessanten Sender- und Empfängerschaltungen. (Minispione) Viele hundert Bilder zeigen Ihnen genau, wie Sie beim Experimentieren vorgehen müssen. IC-Radio, IC-Sender, Antennen, Berechnungsgrundlagen, Tabellen u. v. a. Über 150 Seiten **DM 19,80**

11

IC -Niederfrequenzverstärker , C. Lorenz

Grundlagen der integrierten NF-Verstärker, Berechnung von kompletten IC-NF-Verstärkerstufen. Anwendungsbeispiele mit den interessantesten und gebräuchlichsten Standard IC-NF-Verstärkern wie TBA 800, TBA 830, usw. Printvorlagen, Auswahltabellen, Experimentieranleitungen und Anschlußbilder machen dieses Buch zu einem unentbehrlichen Begleiter für alle, die sich mit NF-Verstärkern beschäftigen wollen. Über 65 Seiten.
DM 9,80

12

BIS BUCH, Beispiele integrierter Schaltungen, H. Bernstein

Auf über 130 Seiten Anwendungsbeispiele mit integrierten Schaltkreisen. Zeitgeber 555, Funktionsgenerator ICL 8038, Opto Elektronik, Operationsverstärker, Analogschalter, Digital-Analog-Wandler, Analoge Rechenbausteine, Schreib-Lese-Speicher (RAM), Festwertspeicher (ROM), Speicherschaltungen, Uhrenbausteine u. v. a. **DM 19,80**

13

HEH, Hobby Elektronik Handbuch , C. Lorenz

Das Schaltungsbuch für jeden Hobbyelektroniker. Schaltbeispiele und Bauanleitungen aus dem gesamten Hobbybereich. Lichtorgeln, Eiswarngerät fürs Auto, Alarmanlagen, Metallsuchgerät, PLL-Schaltungen, Logik-Tester, Funktionsgeneratoren u. v. a. Über 55 Seiten. **DM 9,80**

14

IC - Vergleichsliste , C. Lorenz

Vergleichsliste für digitale und lineare integrierte Schaltkreise.

Standard TTL, Low Power Schottky TTL, C MOS, Triacs Thyristoren, Optoelektronik, Operationsverstärker, Spannungskomparatoren, Spannungsregler, NF-Verstärker u. v. a. Funktionsvergleichsliste CMOS zu TTL. Vergleichstabelle für Transistoren und Dioden sowie Darlingtontransistoren. Eine Vergleichsliste, die man immer wieder braucht. **DM 9,80**

15

Opto -Handbuch, Handbuch für Optoelektronik , C. Lorenz

Das Handbuch für die gesamte Optoelektronik. Eine Einführung und ein ideales Nachschlagwerk. Grundlagen, Definitionen aller Kenngrößen, Opto-Lexikon, Berechnungsgrundlagen, Rechenbeispiele, Schaltbeispiele: Lichtsender, Lichtempfänger, Anzeigen, Infrarot Detektoren, Lichtmeßgerät, Optokoppler, Pegelschalter, Opto-Vergleichsliste. Anschlußbilder wichtiger 7-Segment-Anzeigen u. v. a. Über 106 Seiten. **DM 19,80**

16

C MOS Einführung, Entwurf, Schaltbeispiele, Teil 1 , H. Bernstein

Vom C MOS Gatterbaustein über Schieberegister und Zähler bis hin zum C MOS Schreib-Lesespeicher. Insgesamt werden neunzehn interessante und bekannte C MOS Schaltkreise beschrieben. Zu jedem Bauelement sind genaue Daten, Schaltbild und Anwendungsbeispiele angegeben. Im großen Applikationsteil finden Sie: C MOS-Kippstufen, Addierwerke und Rechenschaltungen, Digital Analog Wandler, Schieberegister für analoge Spannungen, Multiplexsysteme für analoge Signale u. v. a. Eine komplette Einführung und gut geeignet für das Selbststudium der C MOS Technik. Über 140 Seiten. **DM 19,80**

17

C MOS Entwurf und Schaltbeispiele, Teil 2 , H. Bernstein

Fortsetzung von Teil 1. Anwendungsbeispiele mit genauen Schaltungsbeschreibungen und Bauelementeunterlagen. Daten, Anschlußbelegungen weiterer wichtiger hochintegrierter C MOS Elemente. Ein komplettes Arbeits- und Experimentierbuch. C-MOS Uhrenschaltungen, Schieberegisterschaltungen, Parallel-Serien Umsetzung, statische und dynamische Speicherschaltungen, Zählschaltungen, Digital Analog Wandler, Analog Digital Wandler, Digital Voltmeter, I/O Registerschaltungen, Codier und Dekodierschaltungen. RAM und ROM Anwendungen. Über 140 Seiten. **DM 19,80**

18

C MOS Entwurf und Schaltbeispiele, Teil 3 , H. Bernstein

Fortsetzung von Teil 2. Eine sehr umfangreiche Applikationssammlung mit hochintegrierten C MOS Elementen. Rechnerschaltungen, Speicher- und Steuerschaltungen, Multiplex- und Datenbussysteme, Uhrenschaltungen, PLL-Schaltungen, Liquid Cristal Anzeigen und deren Treiberschaltungen, Optoelektronik in Verbindung mit C MOS. Grundlagen, Aufbau und Wirkungsweise der Prozeßrechentechnik, Arithmetische Logische Einheiten (ALU) und andere wichtige Funktionen aus der Prozeßrechentechnik. RAMs, ROMs und FIFO-Speicherschaltungen. Über 140 Seiten. **DM 19,80**

19

IC Experimentier Handbuch - IC -EX, C. Lorenz C. Lorenz

Eine sehr umfangreiche Schaltungssammlung und Bauanleitunssammlung mit neuesten integrierten Bausteinen. Neue, jedoch beim Fachhandel erhältliche Standard ICs. Rechnerschaltungen, Mikroprozessoren, I/O Schaltungen, druckende und anzeigende Rechner, Stoppuhren, Zählerschaltungen, Digitalvoltmeter, professioneller Synthisizer, Hilfsschaltungen für den Elektronik Experimenter, Analog Digital Wandler, Frequenzzähler u. v. a. hochinteressante Bauanleitungen. Viele Schaltungen können auf der IC KIT Experimentierplatine WH–1g aufgebaut werden. Über 120 Seiten. **DM 19,80**

20

Operationsverstärker, Grundlagen und Schaltbeispiele , C. Lorenz

Dieses Buch umfaßt das gesamte Gebiet der linearen Schaltungstechnik und stellt ein in dieser Preislage bisher noch nie dagewesen Nachschlagwerk und Einführungshandbuch dar. Bestens geeignet für das Selbststudium. Nach einer pädagogisch geschickt gemachten Einführung folgen theoretische Arbeitsunterlagen und die zugehörigen Schaltbeispiele mit Daten und Gehäuseanschlüssen. Dieses wertvolle Buch dürfte seinen Platz auch bei Ihren Arbeitsunterlagen finden, und wird dann immer von Nutzen sein, wenn es um die Lösung von nicht routinemäßigen Aufgaben geht. **DM 19,80**

21

Digitaltechnik Grundkurs (TTL -C MOS - MOS und Software) , C. Lorenz

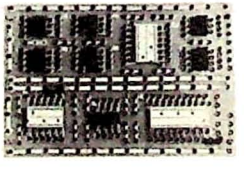
Ein Einführungskurs in die Digitaltechnik für Anfänger und Fortgeschrittene. Ein Fachbuch für den programmierten Selbstunterricht. Der ideale Kurzlehrgang für das Selbststudium. Der Kurs vermittelt Ihnen alle wichtigen Grundkenntnisse vom TTL-Gatter bis zum Mikroprozessor und Lösung von Schaltungsaufgaben durch Software. Viele Versuchsaufbauten und Experimente aus diesem Kurs können auf der IC-KIT Platine WH–1g durchgeführt werden. Grundlagen, Gatter, Zähler, programmierbare Zähler, IC-Tester, Schieberegister, Speicher, Mikroprozessoren u. v. a. Über 130 Seiten. **DM 19,80**

41

Experimentierplatine WH-1g dazu, Best.Nr. 41 DM 79,–

Experimentierplatine WH - 1 g
Abmessungen: 210 mm x 150 mm

Experimentierplatine WH - 1 g
fertig aufgebaut und mit Sockeln bestückt.



22

Mikroprozessoren, Eigenschaften und Aufbau Teil 1 , H. Bernstein

Grundlagen, Eigenschaften und Aufbau von Mikroprozessoren. Organisation von Recheneinheiten und Mikroprogrammen. Programmierung und Klassifizierung von Mikroprozessoren. Ablaufdiagramm, Flußdiagramm. Ein Chip-Technik und Multi Chip-Technik, Transfer- und Sprungfunktionen. Speichertechnik: RAMs, ROMs, FIFO, FILO. Programmierbare logische Arrays (PLA)
Anwendungsbeispiele und Anwendungsbereiche. Über 120 Seiten. **DM 19,80**

23

Elektronik Grundkurs (Kurzlehrgang Elektronik) , C. Lorenz

Eine leichtverständliche und pädagogisch geschickt gemachte Einführung in die Technik der elektronischen Schaltungen. Ein Kurzlehrgang und Schnellkurs zugleich. Aber auch ein recht brauchbares Nachschlagewerk für den fortgeschrittenen Elektroniker. Mit wenig Mühe können Sie sich hier die Grundkenntnisse der elektronischen Schaltungspraxis aneignen. Das Buch schafft die Voraussetzungen für ein erfolgreiches und sicheres Arbeiten mit interessanten Schaltkreisen modernster Technologien. Unentbehrlich für das Experimentieren mit den heutigen modernen hochintegrierten Schaltkreisen. Über 150 Seiten.
DM 9,80

24

Mikrocomputer Technik, Hans Peter Blomeyer-Bartenstein

In diesem Buche finden Sie eine umfassende, einführende und weiterführende Hilfe zum Einstieg in die Mikrocomputertechnik mit vielen Schalt- und Programmierbeispielen. Als praktische Betrachtungsgrundlage dient das supermoderne Microcomputerkonzept Z80A von ZILOG. Das Buch geht auf alle wichtigen Zusammenhänge ein und erklärt diese dem Leser so ausführlich, daß kaum noch Fragen offen bleiben. Über 240 Seiten **DM 29,80**

25

Hobby Computer Handbuch, C. Lorenz

Eine leicht verständliche Einführung in die Microcomputertechnik.

Diese sehr umfangreiche Einführung in die Microcomputertechnik dürfte zu diesem Preise einmalig sein. Auf über 450 Seiten finden Sie: Grundlagen der Computer- und Microcomputer-Technik, Was ist ein Microcomputer?, Microcomputer KITs, Einplatinencomputer, KIM, MIKIT, Z80 KIT, 6800 KIT, NEC 8080 KIT u. v. a. Genaue Beschreibungen der wichtigsten Mikroprozessortypen. Zusammenstellung und Beschreibung der modernen Personal Computer. (IMSAI, CROMEMCO, CAT, OSI, POLY 88 u. v. a.) Kompaktcomputer wie PET und TR-8. Interfactechniken, Ein/Ausgabegeräte, ROMs, RAMs, Programmiergeräte für PROMs. Löschgeräte für EPROMs u. v. a. mehr. Das ideale allumfassende Buch für den Microcomputertechniker. Für Industrieanwendung ebenso geeignet, wie für den Hobby-Computer Fan. Über 450 Seiten **DM 29,80**

26

Mikroprozessor Teil 2, H. Bernstein

Die Fortsetzung unseres ersten so erfolgreichen Buches über Mikroprozessoren. Technologie von Mikroprozessor- und Speicherbausteinen. Festwertspeicher, PROM, REPROM, FIFO, Schieberegister, MPR-Register, ARL-Register, SAR-Register. Aufbau eines Mikroprozessorsystems mit 8080, RAM- und ROM Schnittstellen. Befehlssatz 8080. Über 120 Seiten. **DM 19,80**

27

Mikroprozessor Software Handbuch MSH, C. Lorenz

Grundlagen und Einführung in die Mikroprogrammierung. Grundlagen und Einführung in die wichtigsten Programmiersprachen (BASIC, FORTRAN, ASSEMBLER-Sprachen) Zusammenstellung der wichtigsten Befehlslisten: 8080, Z80, M 6800, National, Fairchild, etc.
Ein Software Handbuch für jeden der mit Mikroprozessoren oder Mikrocomputern zu tun hat. Über 200 Seiten. **Preis DM 29,80**

28

Lexikon und Wörterbuch für Elektronik und Mikroprozessortechnik, LEM, C. Lorenz

Ein Hilfs- und Arbeitsbuch für jeden der sich heute mit der modernsten Elektronik beschäftigt. Viele engl. Ausdrücke werden heute in der Elektronik, Computer- und Mikroprozessortechnik verwendet und oft fehlt uns eine genaueste und präzise Erklärung. Dieses Buch übersetzt Ihnen den englischen Fachausdruck und gibt Ihnen zusätzlich noch eine deutsche Erläuterung und Erklärung dieses Begriffes und was es damit auf sich hat. Ein Lexikon und Wörterbuch in einem einzigen Buch vereinigt. Das Buch, das Sie schon lange gesucht haben. Ca. 250 Seiten. Lieferbar Herbst 1978 **Preis DM 29,80**

29

Mikrocomputer Datenbuch, C. Lorenz

Zusammenstellung der wichtigsten Mikroprozessordaten.

Eine übersichtliche und sehr informative Zusammenstellung der wichtigsten Mikroprozessorbausteine auf dem Markt. 8080A, 8085, 8048, Z80, Z8, 6500, 6800, 2650, 1802, F8, 3870, SC/MP, PACE u. v. a. Daten, Anschlußbilder, wichtige technische und elektrische Daten, Architektur, grundlegende Eigenschaften. Zu jedem Mikroprozessor werden dann auch noch die peripheren Bausteine sowie RAM und ROM Elemente behandelt. Das ideale Handbuch für jeden modernen Elektroniker. Lieferbar Ende 78 **DM 49,–**

30

Aktivtraining - Mikrocomputer (Sammelordner)

Ein Mikrocomputerkurs in Form einer Lehrbriefsammlung.

Das Aktivtraining-Seminar erscheint in einzelnen Lehrbriefen, die in einem formschönen praktischen Plastik-Sammelordner geliefert werden. Die Grundausstattung besteht aus 10 Lehrbriefen und reicht von den elementaren Grundlagen bis hin zu praktischen Beispielen. Weiterhin ist ausführliches Mikrocomputer Datenmaterial im Ordner enthalten. Die Lehrbriefe werden regelmäßig erscheinen und können jederzeit nachbezogen werden. Lieferbar Ende 78 **DM 49,–**

31

57 Programme in BASIC, C. Lorenz

Ein Buch mit technisch-wissenschaftlichen Programmen und einer großen Anzahl von Spielprogrammen in BASIC. (Games) Ein Buch für jeden, der sich mit dem faszinierenden Hobby der Mikrocomputertechnik befassen will. Alle Listings sind in BASIC und können auf den meisten Personal Computer Systemen gefahren werden.
Lieferbar Ende 78 **DM 39,–**

Kurzlehrgang TTL CMOS

TTL – Kurzlehrgang
Der ideale Einführungskurs für das Selbststudium, den Unterricht, die Ausbildung und für den Amateur. Das Buch Nr. 21 zusammen mit unserem IC-KIT Experimentierplatinenbausatz WH-1g bilden die ideale Grundlage für diese TTL-Einführung. Kreuzen Sie auf der Bestellkarte einfach die Bestellnummern 21 und 41 an, Sie erhalten dann umgehend diesen wertvollen Kurzlehrgang zugeschickt.
Best. Nr. 21/41 DM 98,80

21/41

CMOS – Kurzlehrgang
Der ideale Einführungskurs in die moderne C-MOS Technik. Grundlagen, Entwurf, Versuchsschaltungen. Das Buch Nr. 16 zusammen mit der IC-KIT Experimentierplatine WH-1g bilden die ideale Grundlage für diesen interessanten Kurs. Kreuzen Sie auf der Bestellkarte einfach die Bestellnummern 16 und 41 an. Sie erhalten dann umgehend diesen C MOS-Kurs.
Best. Nr. 16/41 DM 98,80

16/41

Abs.
Name

Straße:

Ort:

Bestellkarte

Den Gesamtbetrag für die umseitig bestellten Bücher bitte ich per Nachnahme zu berechnen/habe ich heute auf Ihr Postscheckkonto München 15994-807 überwiesen. Nichtzutreffendes bitte streichen.

○ Bitte liefern Sie mir auch die Experimentierplatine IC - KIT WH-1 g für 14,16,24,28 und 40 polige DIL IC-Gehäuse. Bausatz komplett: DM 79.–

○ Bitte liefern Sie von jedem neu erscheinenden Buch ein Muster per Nachnahme. (Rückgaberecht)

Ing.
W. Hofacker GmbH
Verlag
Postfach 437
8000 München 75

Neue Bücher über Mikro-prozessoren

Programming Microprocessors

Zahlensysteme, Flußdiagramme, Adressierungsarten, Assemblersprachen, Programmierhilfen u. v. a. Das ideale Microprozessorbuch mit Schwerpunkt Software. Über 270 Seiten.

Best. Nr. 985 DM 35,–

Microprocessor/Microprogramming Handbook.

Das praktische Handbuch für jeden der sich mit Mikroproz. beschäftigen will. Über 290 Seiten.

Best. Nr. 785 DM 35,–

Beginners Guide to Computer-Programming.

Die ideale Einführung in die Computerprogrammierung. Gut geeignet für das Selbststudium. Über 480 Seiten.

Best. Nr. 574 DM 39,–

Computer Programming Handbook

Eine genaue Einführung in die Programmierung und EDV mit ausgeführten Beispielen. Der ideale "Ein Buch Computerkurs". Über 510 Seiten.

Best. Nr. 752 DM 45,–

Modern Guide to Digital Logic Processors, Memories, Interfaces.

Speicherschaltungen, Interfaceschaltungen, ECL, Schottky TTL u. v. a. mehr. Über 290 Seiten.

Best. Nr. 709 DM 35,–

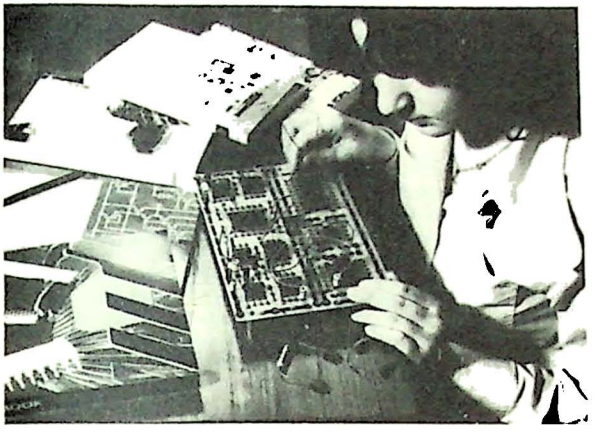
Master Handbook of 1001 practical Electronic Circuits.

Ein Schaltungsbuch mit über 1000 Schaltbeispielen mit kurzen Beschreibungen. Über 600 Seiten.

Best. Nr. 800 DM 49,–

Best. Nr.	Stck.	BUCHTITEL BEZEICHNUNG	à DM	ges. DM
1		TBB – Handbuch Band 1	19.80	
2		TBB – Handbuch Band 2	19.80	
3		Elektronik im Auto	9.80	
4		IC – Handbuch, TTL, CMOS, Lin.	19.80	
5		IC – Datenbuch, TTL, CMOS, Lin	9.80	
6		IC – Schaltungen, TTL, CMOS	9.80	
7		Elektronik Schaltungen	5.–	
8		IC – Bauanleitungshandbuch	19.80	
9		Feldeffekttransistoren	5.–	
10		Elektronik und Radio 4. Auflage	19.80	
11		IC – NF – Verstärker	9.80	
12		Beispiele integrierter Schaltungen	19.80	
13		HEH, Hobby Elektronik Handbuch	9.80	
14		IC – Vergleichsliste	9.80	
15		Optoelektronik Handbuch	19.80	
16		CMOS Teil 1, Einführung	19.80	
17		CMOS Teil 2, Schaltbeispiele	19.80	
18		CMOS Teil 3, Schaltbeispiele	19.80	
19		IC – Experimentier Handbuch	19.80	
20		Operationsverstärker	19.80	
21		Digitaltechnik Grundkurs	19.80	
22		Mikroprozessoren 2 Auflage	19.80	
23		Elektronik Grundkurs	9.80	
24		Mikrocomputer Technik	29.80	
25		Hobby Computer Handbuch	29.80	
26		Mikroprozessor Teil 2	19.80	
27		Mikroprozessor Software Handbuch	29.80	
28		Lexikon I Wörterbuch	29.80	
29		Microcomputer Datenbuch	49.80	
30		Aktivtraining Microcomputer	49.80	
31		57 Programme in BASIC	39.–	
32		Circuits Digital Et Pratique	19.80	
33		Microcomputer Programmierbeispiele	19.80	
41		IC KIT Experimentierplatine WH-1 g	79.–	
105/1		TTL Experimentierbuch	5.–	
106/1		CMOS-Experimentierbuch	5.–	

IC KIT Experimentier-platine



Mit der IC-KIT Experimentierplatine wird jetzt das Aufbauen von Schaltungen mit ICs, Transistoren, Widerständen und Kondensatoren zum Kinderspiel. Einfach, schnell und ohne Mühe lassen sich mit dieser hochwertigen Epoxy Versuchsplatine selbst die komplizierten Versuchsschaltungen aufbauen.
Die IC KIT Experimentierplatine Typ WH-1 g eignet sich zum Aufbau von Uhren-, Rechner- und Mikroprozessorschaltungen besonders gut, da sie neben 14,16,24 und 28 poligen DIL-Sockeln auch einen 40 poligen Stecksockel enthält.

Der Einsatz von Adaptern ermöglicht auch das Stecken von diskreten Bauelementen. Weiterhin können Sie sich mit diesen Adaptern selbst kleine Funktionsgruppen aufbauen, die Sie immer wieder verwenden können, zum Beispiel Taktgeneratoren, Logiktester, Tastenentprellschaltungen etc. Der Schaltungsaufbau wird durch das Stecken aller Bauteile und Verbindungsleitungen wesentlich vereinfacht. Die einzelnen Elemente werden nicht beschädigt und bleiben frei von Lötzinn. Sie können immer wieder verwendet werden. Sie sparen Zeit und Geld!

41

Der Bausatz IC KIT WH-1 g enthält: 1 Experimentierplatine in stabiler Epoxy-Ausführung mit den Abmessungen 210 x 150 mm fertig gebohrt und Leiterbahnen verzinnt. Die Bestückungsseite ist mit einem kratzfesten Lack überzogen und bedruckt. Sie können so die gesuchten Anschlüsse beim Experimentieren immer leicht und schnell finden. 340 Messingbuchsen, verschiedenfarbige Stecker mit Querloch (2,6 mm ϕ), Metallfüße und sämtliche Sockel.

Bestell Nr. 41 — Preis DM 79.–

41$_1$

Adapter zum Stecken von diskreten Bauelementen, 5 Stück in einem Beutel verpackt. Die diskreten Bauteile werden hier einmal angelötet und können dann immer wieder verwendet werden. Einfachste Handhabung.

Bestell Nr. 41/1 — Preis DM 14,90

48

5V Netzgerät stabilisiert für die IC KIT Experimentierplatine Typ WH-1 g. Bausatz komplett mit Netztransformator, Platine, allen Bauelementen, Netzkabel etc. Maximale Welligkeit 1 %, TK = 0,5 mV pro °C, Strombegrenzung. Maximaler Ausgangsstrom 650 mA. Ideal für alle TTL und CMOS Experimente.
Bestell Nr. 48 — Preis DM 39.–

32

Circuits Digital Et Pratique, C. Lorenz

Dieses Buch ist eine französische Übersetzung unserer beiden Bücher Nr. 6 und Nr. 13 (IC-Schaltungen und Hobby Elektronik Handbuch HEH)
Ca. 100 Seiten

DM 19,80
FF 39,90

33

Mikrocomputer Programmierbeispiele für 265 Dr. J. Hatzenbichler

Eine Einführung in die Programmierung von Mikrocomputern an Hand des Prozessors 2650 von Signetics. Viele Programmierbeispiele in Maschinensprache, die Sie auf einem preiswerten Mikroprozessorsystem MIKIT 2650-P2 ausführen können. Zeitschleifenprogramme, Blinkschaltung, Lauflicht, Stufenzähler, Elektronischer Würfel, Stoppuhr, Reaktionszeittester, Computer Musik Programm u. v. a. Jeder Befehl wird genau erläutert und an Hand eines ausführlichen Flußdiagrammes erklärt. Jedes Programm liegt in Form eines Computer-Listings vor. Jedes Programm ist genauestens beschrieben. Sie können so auf einfache Weise die Zusammenhänge erkennen und erwerben damit die Grundkenntnisse zur Erstellung Ihrer eigenen Programme. Sie erkennen, wie man mit nur einer Schaltung (Mikrocomputer) unendlich viele praktische Anwendungsschaltungen realisieren kann. Zu diesem Buche ist auch ein komplett aufgebautes und getestetes Mikrocomputersystem erhältlich, auf dem Sie alle beschriebenen Programme selbst ausführen können. Über 120 Seiten

DM 19,80

105
1

TTL - Experimentierbuch

Eine kleine Einführung in die Digitaltechnik

Grundlagen der Digitaltechnik kurz erklärt. Bauanleitung für einen praktischen Logik-Tester. Viele Experimente und Anwendungsschaltungen mit dem TTL Gatterbaustein 7400. Das ideale Einführungsbuch in die Digitaltechnik.

DM 5,–

106
1

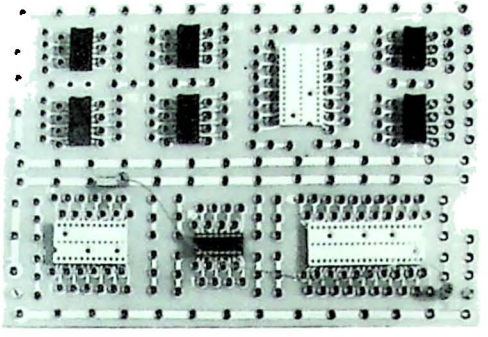
CMOS - Experimentierbuch

Eine kleine Einführung in die CMOS Schaltungstechnik

CMOS Grundlagen, Behandlungshinweise für CMOS Bausteine, Zusammenstellung der wichtigsten CMOS Bausteine und deren Anschlußbilder. Viele praktische CMOS Schaltbeispiele. Ideal für jeden Elektroniker.

DM 5,–

IC KIT Experimentierplatte



Experimentierplatine IC-KIT WH-1g fertig aufgebaut. Abmessungen: 210mmx150mm

Mit der IC - KIT Experimentierplatine Typ WH-1 g wird das Aufbauen von Schaltungen mit integrierten Schaltungen Transistoren, Widerständen und Kondens. jetzt zum Kinderspiel. Einfach, schnell und ohne Mühe lassen sich mit dieser hochwertigen Epoxy-Versuchsplatine selbst Uhren-, Rechner-, Mikroprozessor-Schaltungen aufbauen.
Vereinfachter Schaltungsaufbau durch Stecken der Bauteile und sämtlicher Verbindungsleitungen. Alle Bauteile bleiben ohne Lötzinn und völlig unbeschädigt. Sie können daher wiederverwendet werden und man spart dadurch mit der Zeit erhebliche Geldbeträge ein.

Der Bausatz WH-1g enthält: 1 Experimentierplatine 210mm x 150mm in stabiler Epoxyausf., fertig gebohrt, Leiterbahnen verzinnt, Bestückungsseite bedruckt und kratzfest lackiert, 340 Messingbuchsen zum Einsetzen und verlöten ! Kein Nieten erforderlich!, diverses Steckermaterial 2,6 mm Ø mit Querloch in verschiedenen Farben, Sämtliche Sockel(40, 28, 24, 16 und 14 polige) sowie Metallfüße mit Befestigungsschrauben. Preis: DM 79.-
Zu beziehen durch Ihren Elektronik Fachhändler oder durch den Verlag.

Bei Betrieb von Sende-und Empfangsanlagen gelten die Bestimmungen der Deutsch. Bundespost.
Es kann keine Gewähr übernommen werden, daß die in diesem Buche verwendeten Angaben, Schaltungen, Warenbezeichnungen, Warenzeichen und Begriffe frei von Schutzrechten sind. Alle Angaben erfolgen nur für Amateurzwecke. Nachdruck und öffentliche Wiedergabe nur mit unserer Genehmigung. Irrtum , sowie alle Rechte vorbehalten.
Copyright by Ing. W. Hofacker GmbH 8000 München 75 .Gedruckt in Deutschland. Printed in Germany.

ING. W. HOFACKER

GMBH

FACHBUCHVERLAG · ELEKTRONIK
8 MÜNCHEN 75
POSTFACH 437
Tel. 08024/7331

Verlagsprogramm

Übersicht lieferbarer Bücher

Bestell Nr.	ISBN	Verfasser	Titel	Preis DM
1	3-921682-01-0	Hofacker	Transistor Berechnungs- und Bauanleitungshandbuch, Band 1	19,80
2	3-921682-02-9	Hofacker	Transistor Berechnungs- und Bauanleitungshandbuch, Band 2	19,80
3	3-921682-03-7	Gebauer	Elektronik im Auto	9,80
4	3-921682-04-5	Lorenz	IC-Handbuch, TTL, CMOS, Linear	19,80
5	3-921682-05-3	Steinbach	IC-Datenbuch, TTL, CMOS, Linear	9,80
6	3-921682-06-1	Steinbach	IC-Schaltungen, TTL, CMOS, Linear	9,80
7	3-921682-33-9	Hofacker	Elektronik Schaltungen	5, –
8	3-921682-08-8	Lorenz	IC-Bauanleitungs-Handbuch	19,80
9	3-921682-09-6	Lorenz	Feldeffekttransistoren	5, –
10	3-921682-34-7	Lorenz	Elektronik und Radio, 4. Auflage	19,80
11	3-921682-11-7	Lorenz	IC-NF Verstärker	9,80
12	3-921682-12-6	Bernstein	Beispiele Integrierter Schaltungen (BIS)	19,80
13	3-921682-13-4	Lorenz	HEH, Hobby Elektronik Handbuch	9,80
14	3-921682-14-2	Lorenz	IC-Vergleichsliste	9,80
15	3-921682-15-0	Lorenz	Optoelektronik Handbuch	19,80
16	3-921682-16-9	Bernstein	CMOS Teil 1 Einführung Entwurf, Schaltbeispiele	19,80
17	3-921682-17-7	Bernstein	CMOS Teil 2 Entwurf und Schaltbeispiele	19,80
18	3-921682-18-5	Bernstein	CMOS Teil 3 Entwurf und Schaltbeispiele	19,80
19	3-921682-19-3	Lorenz	IC-Experimentier Handbuch	19,80
20	3-921682-20-7	Lorenz	Operationsverstärker	19,80
21	3-921682-21-5	Lorenz	Digitaltechnik Grundkurs	19,80
22	3-921682-22-3	Bernstein	Mikroprozessoren, Eigenschaften und Aufbau 2. Aufl.	19,80
23	3-921682-23-1	Lorenz	Elektronik Grundkrus Kurzlehrgang Elektronik	9,80
24	3-921682-35-5	Hans Peter Blomeyer-Bartenstein	Mikrocomputer Technik	29,80
25	3-921682-25-8	C. Lorenz	Hobby Computer Handbuch	29,80
26	3-921682-26-8	H. Bernstein	Mikroprozessor Teil 2	19,80
27	3-921682-27-4	C. Lorenz	Mikrocomputer Software Handbuch	29,80
28	3-921682-28-2	C. Lorenz	Lexikon + Wörterbuch für Elektronik und Mikroprozessortechnik LEM	29,80
29	3-921682-29-0	C. Lorenz	Mikrocomputer Datenbuch	49,80
30	3-921682-30-4	C. Lorenz	Aktivtraining-Mikrocomputer (Ordner)	49,80
31	3-921682-31-2	C. Lorenz	57 Programme in BASIC (Games)	39, –
32	3-921682-32-0	C. Lorenz	Circuits Digital Et Pratique	
33	3-921682-36-6	Dr. Hatzenbichler	Microcomputer Programmie	
41	–	–	Experimentierplatine für 14, 40 polige DIL IC's	
105/1	–	–	TTL-Experimentierbuch	
106/1	–	–	CMOS-Experimentierbuch	

Ing. W. Hofacker GmbH Ve
8 München 75